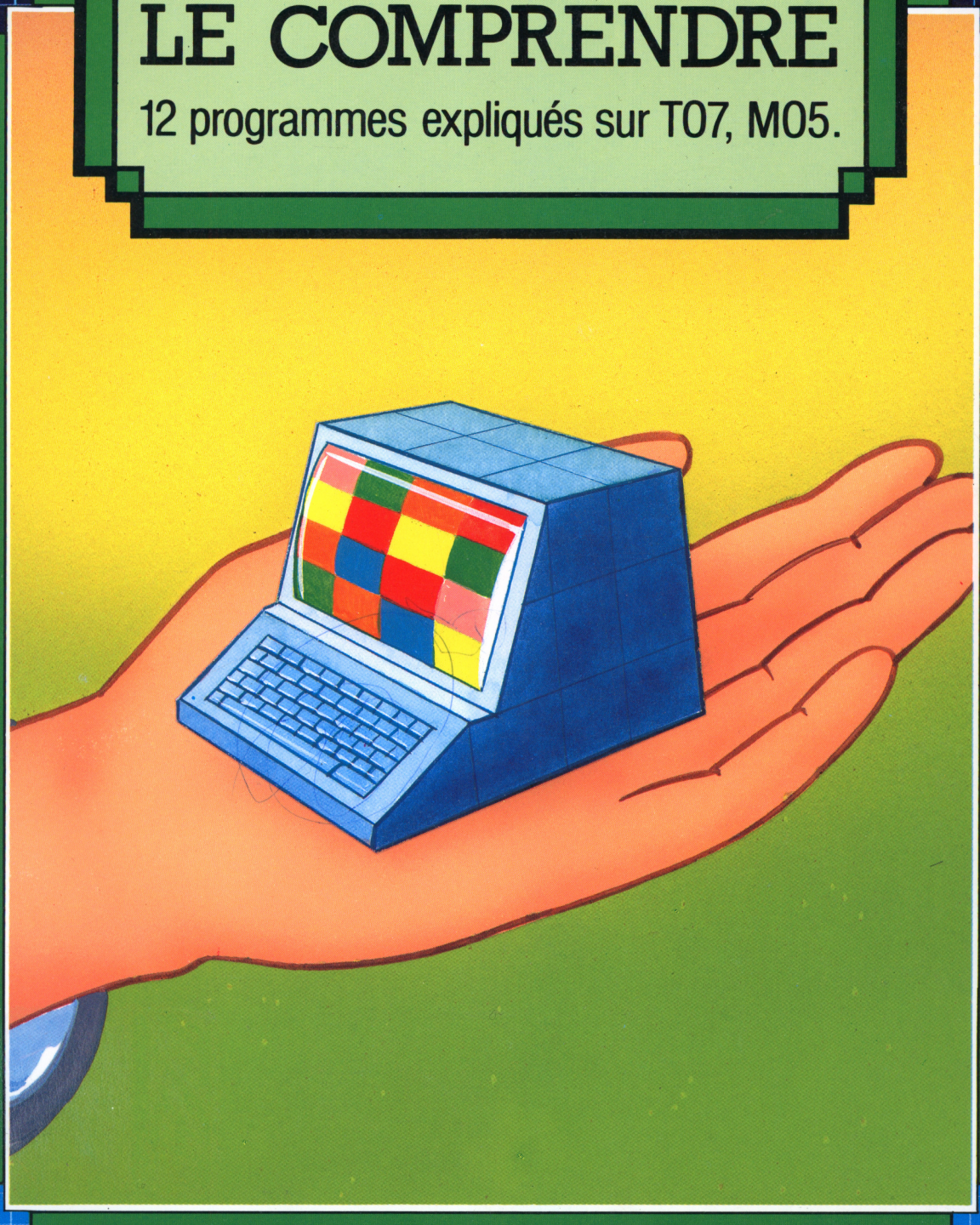
RENDEZ-VOUS MICRO

POUR MIEUX LE COMPRENDRE

12 programmes expliqués sur T07, M05.



FOUCHER

RENDEZ-VOUS MICRO

POUR MIEUX LE COMPRENDRE

RENDEZ-VOUS MICRO

Rolf Lohberg • Theo Lutz

POUR MIEUX LE COMPRENDRE

12 programmes expliqués sur T07, M05.

Illustrations : Gerhard Utecht

MICRO

LES ÉDITIONS FOUCHER
128 rue de Rivoli 75001 Paris

Sommaire

Avant-propos 5
Le BASIC en douze étapes 6
Qu'est-ce que le BASIC ? 8
Renards et lièvres 10
Le calendrier 12
Deviner des nombres tirés au hasard 14
Répertoire téléphonique 16
L'ordinateur devine des nombres 18
Le jeu de NIM 20
Simulation 22
Peut-on passer des nombres au crible ? 24
Le carré magique 26
Jeu de la vie 28
Cinq fois l'infini : suites et séries 30
L'ordinateur écrivain : est-ce possible ? 32
Comment travailler sur un programme 34
Une bonne idée 36
Index 37

Traduit de l'allemand par Frédéric RUTKOWSKI

ISBN 2-216-00931-8

Imprimé au Portugal

Avant-propos

De nombreuses personnes possèdent un ordinateur domestique, mais seule une minorité peut se vanter d'avoir une réelle bibliothèque de programmes. Peut-être faites-vous partie de ceux qui recherchent de nouveaux programmes pour leur ordinateur ? Ce livre est fait pour vous. Vous y trouverez douze programmes accompagnés de leur mode d'emploi.

Ces logiciels ont été testés et ils utilisent un langage BASIC élémentaire. En conséquence, vous devez pouvoir les taper sur n'importe quelle machine sans rencontrer de difficulté majeure. Ils ont d'ailleurs fonctionné sur un ordinateur de poche. Dans ce genre de machines, on peut utiliser 26 variables au maximum, chacune portant un nom de 7 caractères ; il reste alors 1 424 pas de programme. Le nombre de variables peut augmenter, mais, dans ce cas, la place réservée au programme diminue. Nos programmes tiennent compte de ces impératifs, et vous serez étonné de tout ce que l'on peut faire avec de si petites machines. Les listings de ce livre sont ceux correspondant à la version THOMSON.

Bien évidemment, ces douze programmes fonctionnent également sur un matériel plus sophistiqué. Vous pourrez alors améliorer et compléter chacun de nos logiciels en vous inspirant des remarques et des conseils que nous vous donnons dans ce livre. Nous vous expliquons également chaque programme en détail. Ce livre constitue donc votre outil de travail.

Si vous n'êtes pas un expert en BASIC, ce n'est pas grave ! Certes, quelques notions de ce langage vous faciliteraient la tâche. De toute façon, nous avons supprimé toute complication et autre "tour de magie" qui rendraient peut-être les programmes plus élégants, mais en compliqueraient inutilement la compréhension. Lorsque vous serez bien familiarisé avec nos programmes, vous pourrez les modifier vous-même. Par exemple : notre programme simulant les flux de la circulation routière comporte seulement deux carrefours ; vous pourrez le modifier pour concevoir le réseau entier de la ville de votre choix. En pratiquant de la sorte, vous approfondirez grandement votre connaissance du BASIC.

Maintenant, nous pouvons commencer. Amusez-vous en utilisant nos programmes sur votre ordinateur personnel !

Le BASIC en douze étapes

Le but de ce livre est de vous fournir douze programmes déjà réalisés, et qu'il vous suffit de taper. Toutefois, il n'est pas évident qu'un programme fonctionne du premier coup, car une faute de frappe suffit à provoquer une erreur de programmation. Nous vous garantissons que ces logiciels fonctionnent réellement sur les ordinateurs THOMSON MO5 et T07/70. Pour que vous trouviez le plus de plaisir possible à la lecture de ce livre, nous avons mélangé les genres. Enfin, notre objectif est de vous permettre de maîtriser ces programmes. Le choix du BASIC comme langage de programmation n'est pas un hasard ; ce langage est compris par la plupart des ordinateurs domestiques ainsi que par de nombreuses calculatrices programmables.

Structure du livre

Ce livre est votre manuel de travail. Il vous indique comment résoudre les problèmes liés à la programmation et vous propose des solutions astucieuses. Vous avez besoin d'un ordinateur connaissant le langage BASIC. Si vous êtes surpris par certains aspects simplistes de nos programmes, n'oubliez pas que nous avons agi de façon à rendre ces logiciels facilement compatibles avec n'importe quel ordinateur. Nous avons donc particulièrement veillé à rester clairs et compréhensibles.

Vous trouverez un logiciel par double page. Tout d'abord, un texte d'introduction définit le but du programme. L'idée maîtresse est ensuite développée. Puis les grandes phases du programme sont analysées et commentées. Des données-test vous permettront de vérifier la validité des résultats de votre programme, et donc d'être sûr de son bon fonctionnement. Enfin, le listing apparaît tel que nous l'avons édité sur l'imprimante.

Procédure

Quand vous souhaitez utiliser un programme, tapez-le au clavier de votre ordinateur. N'oubliez pas d'effacer la mémoire de votre ordinateur en tapant tout d'abord les commandes NEW (effacement de programme) et CLEAR (effacement de variables).

Il est possible que vous ayez fait une faute de frappe ou introduit une erreur malencontreuse. Afin de les dépister, vous devez tester le programme en utilisant pour cela les données particulières que nous vous indiquerons.

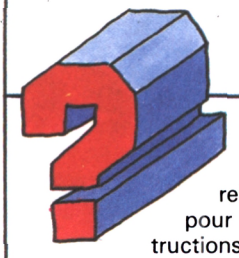
Conservez dans votre bibliothèque une cassette ou une disquette contenant le programme définitif, testé. Cela vous évitera de le retaper chaque fois que vous en aurez besoin.

Aide à la mise au point

Presque tous les ordinateurs, même les calculatrices programmables, possèdent quelques commandes facilitant la recherche et la localisation des erreurs.

Certains BASIC comprennent l'instruction DEBUG, ce qui signifie "ôter les erreurs". Le programme s'exécute ligne par ligne en s'arrêtant à la fin de chaque ligne. Vous pouvez ainsi examiner les valeurs de chaque variable et suivre très précisément le cheminement du programme. Si votre ordinateur ne comprend pas cette instruction, vous pouvez insérer l'instruction STOP, précédée d'un numéro de ligne, autant de fois que vous le désirez dans le corps du programme. L'exécution s'arrêtera ; vous pourrez vérifier le contenu des variables ; puis vous relancerez l'exécution en tapant (sans numéro de ligne) CONT.

Beaucoup de BASIC reconnaissent la commande TRON (ou TRACE). Tapez TRON puis lancez l'exécution du programme. Tous les numéros des lignes exécutées apparaîtront à l'écran.

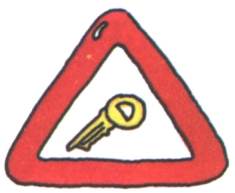


Les langages BASIC

Le BASIC est un langage de programmation relativement ancien. Il a été conçu en 1963 pour être utilisé par des débutants ; ses instructions étaient donc simples et peu nombreuses.

Avec l'avènement des calculatrices programmables et des microprocesseurs, le BASIC s'est vite répandu, car il convenait parfaitement à ces petites machines. En effet, les mots BASIC sont plus facilement compris par le microprocesseur que d'autres langages de programmation.

Avec l'essor de l'informatique et des ordinateurs, le BASIC s'est développé et continue à se développer. Chaque fabricant a ajouté ses propres instructions, si bien qu'aujourd'hui, s'il subsiste un "noyau" commun à tous les BASIC, chaque machine présente des variantes. Il arrive fréquemment qu'un programme fonctionnant sur un ordinateur ne puisse être exécuté sur un autre.



Devons-nous tenir compte des différences entre les BASIC ?

Le fait de rencontrer un langage BASIC particulier peut expliquer qu'un programme correctement tapé ne fonctionne pas sur votre propre ordinateur. Reportez-vous au manuel accompagnant votre machine ; vérifiez que l'instruction refusée par votre ordinateur existe bien : peut-être est-elle simplement mal orthographiée ou manque-t-il une parenthèse ? Si l'instruction incriminée n'existe pas, vous trouverez certainement un équivalent (par exemple, l'instruction CLEAR s'écrit CLR sur certaines machines ; le résultat est le même). En tapant nos programmes, les problèmes que vous pourrez rencontrer seront minimes, car nous avons veillé à éliminer les mots clés ésotériques.

Le manuel de l'utilisateur

Tout ordinateur et tout langage de programmation sont accompagnés d'un manuel destiné à l'utilisateur. Certes, quelques personnes sont prêtes à découvrir d'elles-mêmes toutes les erreurs possibles. Mais elles gaspillent ainsi beaucoup de leur temps et il est préférable d'étudier le manuel en profondeur. La qualité du mode d'emploi est importante ; d'elle dépend le résultat que vous atteindrez avec votre machine.

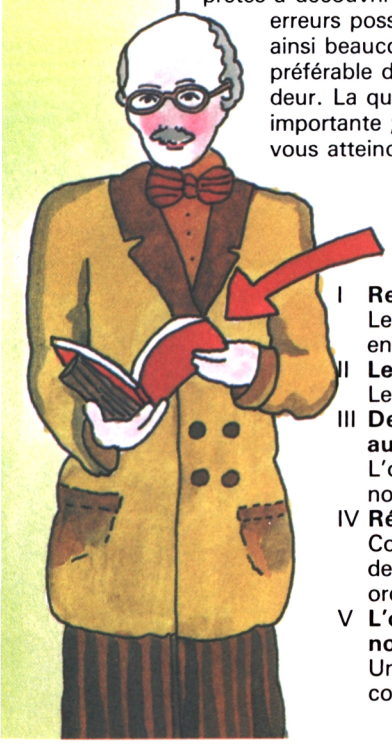


Table des matières

		Pages
I	**Renards et lièvres** — Le développement de deux races ennemies.	10-11
II	**Le calendrier** — Le squelette d'un calendrier annuel.	12-13
III	**Devinez des chiffres tirés au hasard** — L'ordinateur tire au hasard un nombre que vous devez trouver.	14-15
IV	**Répertoire téléphonique** — Comment enregistrer et retrouver des numéros de téléphone dans un ordinateur.	16-17
V	**L'ordinateur devine des nombres** — Un programme qui gagne à tous les coups.	18-19
VI	**Le jeu du NIM** — Un jeu réel avec l'ordinateur.	20-21
VII	**Simulation** — Comment traiter la circulation automobile par ordinateur.	22-23
VIII	**Peut-on passer des nombres au crible ?** — On découvre les nombres premiers.	24-25
IX	**Le carré magique** — Un ordinateur peut le calculer à votre place.	26-27
X	**Le jeu de la vie** — L'ordinateur prévoit l'évolution d'une race.	28-29
XI	**Cinq fois l'infini** — Suites et séries, trouver Pi.	30-31
XII	**L'ordinateur écrivain : est-ce possible ?** — Il construit des textes ressemblant à des poèmes.	32-33

Qu'est-ce que le BASIC ?

Le BASIC est un langage de programmation de haut niveau. Il s'adresse aussi bien à l'utilisateur averti qu'au programmeur débutant. D'où son nom : BASIC, qui veut dire "fondamental". Ce langage est constitué de mots simples, mais il ne sera pas toujours facile à apprendre, surtout lorsque ce que l'on voudra représenter en BASIC ne sera pas formulé d'une manière simple.
Ce livre n'a pas pour objet de vous initier au BASIC. Si ce sujet vous intéresse, vous pouvez lire d'autres ouvrages, notamment "Voyage en BASIC" publié dans la même collection.

Ces deux pages sont là uniquement pour vous rappeler quelques notions essentielles de ce langage : c'est "le BASIC en un clin d'œil". Nous détaillons également les mots clés qui interviennent dans nos programmes.

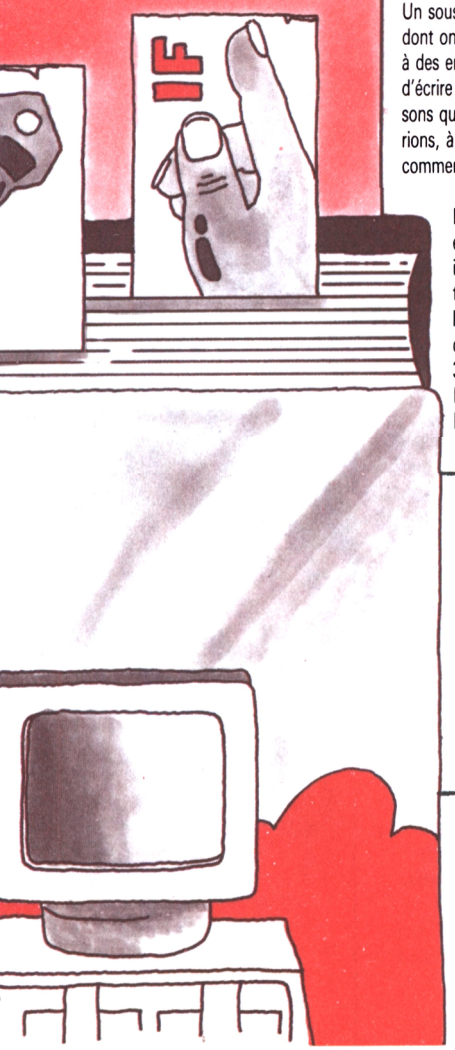
Programmes, commandes et instructions
Un programme BASIC est une suite d'ordres successifs. Chaque ordre est écrit sur une ligne commençant par un numéro de ligne. Les instructions sont exécutées dans l'ordre croissant des numéros de ligne. Pour permettre d'insérer ultérieurement des instructions sans réécrire tout le programme, on attribue généralement des numéros de lignes allant de 10 en 10, l'ordinateur reclassant automatiquement les lignes d'instruction par ordre croissant. Un ordre se décompose en un ou plusieurs mots clés. Ce sont des instructions BASIC destinées à l'ordinateur. Ces mots clés sont en anglais ; il faut connaître leur signification et leur mode d'emploi.
Enfin, un ordre est souvent accompagné de "données". Ces dernières sont appelées "variables" quand leur valeur peut changer durant le programme. Elles sont appelées "constantes" quand leur valeur est fixe.

Les données représentent des nombres ou des caractères
Les données sont constituées de dessins, d'informations ou de valeurs. Elles peuvent être "numériques" (elles représentent alors des chiffres ou des nombres) ou "alphanumériques" (elles contiennent des lettres, des chiffres ou des signes de ponctuation, le tout formant ce qu'on appelle une chaîne de caractères que l'on place entre guillemets). Les valeurs numériques sont écrites normalement, sans guillemets ; toutefois, la virgule est remplacée par un point.

Les noms de données
Les données (numériques ou alphanumériques) dont les valeurs peuvent changer au cours du programme sont appelées des "variables". Chaque variable numérique reçoit un nom composé d'une ou plusieurs lettres auxquelles peuvent être accolés des nombres (exemple : K15, F ou B3). Les noms des variables alphanumériques (ou chaînes de caractères) sont composés de la même manière, mais se terminent toujours par le signe $ (exemple : A$, G18$ ou B3$). Dans nos programmes, ces règles sont scrupuleusement respectées.

Formules, expressions et instruction LET

Une "expression" est une ligne composée de noms de variables, de constantes, de parenthèses et des signes arithmétiques : + (addition), – (soustraction), * (multiplication) et / (division). Généralement, tous ces noms et signes se suivent sans espace. Toutes les variables utilisées dans une expression doivent auparavant avoir reçu une valeur dans le cours du programme (sinon, la valeur 0 est prise par défaut, mais ce n'est pas une bonne manière de programmer). Ainsi, si nous voulons attribuer la valeur –3.14 à la variable M, nous écrivons (par exemple pour la ligne 150) : 150 M = –3.14
Le signe = utilisé dans cette ligne n'indique pas que M est égal à –3.14 mais qu'on ATTRIBUE (ou encore, qu'on assigne) la valeur –3.14 à la variable M. On peut tout à fait utiliser une expression à droite du signe =. L'ensemble de la ligne constitue une formule. Pour éviter de confondre égalité et attribution, on utilise l'instruction LET en début de ligne. Toutefois, sur de nombreux ordinateurs modernes, cette instruction est facultative.



Boucles et instructions FOR, TO, NEXT

Quand une instruction ou une suite d'instructions doit s'exécuter plusieurs fois de suite dans le corps d'un programme, on la place dans une "boucle". Une boucle qui doit s'exécuter dix fois commence ainsi :

250 FOR N = 1 TO 10

et se termine par :

350 NEXT N

(250 et 350 sont des numéros de lignes arbitraires choisis à titre d'exemple). Les lignes placées entre 250 et 350 s'exécuteront dix fois ; N sera incrémenté de 1 à chaque fois.

L'instruction de saut inconditionnel GOTO

Supposons que l'on écrive : 750 GOTO 100
Lorsque le programme exécute la ligne 750, il comprend que la prochaine ligne à exécuter sera la ligne 100. Cela s'appelle un saut inconditionnel.

Le sous-programme : instructions GOSUB et RETURN

Un sous-programme est une série d'instructions dont on a besoin plusieurs fois dans le programme, à des endroits différents. Son utilisation évite d'écrire plusieurs fois la même séquence. Supposons que dans notre programme principal nous désirions, à la ligne 310, appeler un sous-programme commençant à la ligne 900 ; nous écrirons :

310 GOSUB 900

Le programme saute alors à la ligne 900 et exécute, à partir de cet endroit, toutes les instructions ligne par ligne jusqu'à l'instruction RETURN. "RETURN" commande à l'ordinateur de retourner d'où il vient, c'est-à-dire à la première ligne placée après la ligne 310.
Les lignes placées entre 900 et l'instruction RETURN constituent un sous-programme.

Commentez votre programme avec "REM"

Il est souvent utile de pouvoir insérer dans un programme des commentaires, des remarques, des explications. Il suffit pour cela de placer le mot REM (comme REMarque) après le numéro de ligne, puis votre commentaire.
Tous les caractères du clavier sont utilisables. En voici un exemple :
390 REM C'EST LA SUITE DE LA PARTIE B DU PROGRAMME

STOP et END

END signale la fin du programme principal pour éviter que ce programme, en se terminant, passe sur les sous-programmes placés après.
STOP est utilisé comme arrêt temporaire lors de la mise au point. L'exécution reprend si l'on tape CONT au clavier, ce qui n'est pas possible avec l'instruction END.

Tests et décisions : l'instruction IF ... THEN

Il arrive fréquemment que l'on ait à prendre une décision en fonction du contenu d'une variable. Par exemple, si l'on veut retirer une somme d'un compte bancaire, cela n'est possible que s'il reste suffisamment d'argent sur le compte. La ligne :
520 IF (SOLDE–RETRAIT) = 0 THEN GOTO 730
permet à l'ordinateur de vérifier SI (IF) le solde après retrait est négatif ou nul. Si tel est le cas, l'exécution passe à la ligne 730 ; sinon, elle continue normalement à la ligne 521.

Saisie de données : l'instruction INPUT

L'instruction INPUT demande à l'ordinateur de scruter le clavier et de lire tous les caractères pouvant être tapés par l'utilisateur jusqu'à ce qu'il appuie sur la touche RETURN.
470 INPUT "S.V.P. TAPEZ LA SOMM";B
Arrivé à la ligne 470, le programme attend que l'utilisateur ait tapé une valeur numérique suivie de RETURN. cette valeur est placée dans la variable B et le programme continue.

Affichage de messages et de résultats : l'instruction PRINT

Ecrivons la ligne suivante :
770 PRINT "SOMME =";B
L'ordinateur affiche le message placé entre guillemets puis, sur la même ligne, le contenu de la variable B.

Les différents langages BASIC

Aujourd'hui, il existe un très grand nombre de langages BASIC. Cela constitue un véritable défi : quand le programme, une fois tapé, ne fonctionne pas, cela peut provenir d'une faute de frappe, d'une faute de programmation, d'une erreur du programme, d'une instruction mal comprise ou d'une particularité du BASIC de l'ordinateur utilisé. Il existe deux arbitres : le manuel de l'utilisateur (ou mode d'emploi) et l'ordinateur. Surtout, ne désespérez pas, mais transformez-vous en détective. Trouvez l'erreur ! Faites très attention aux détails.

Renards et lièvres

Trop de renards, cela entraîne la disparition des lièvres. Dans les vieux livres d'école, on trouve des exercices de calcul comme celui-ci : connaissant la vitesse du lièvre et celle du renard, combien de temps mettra le renard pour rattraper le lièvre ? Dans les livres modernes, d'autres problèmes sont posés : sur une île vivent des lièvres et des renards. La nourriture est surabondante pour les lièvres, ce qui leur permet de se multiplier chaque année suivant un pourcentage connu. Mais sur cette île vivent également des renards. Bien sûr, ils dévorent les lièvres. Lorsque les lièvres sont nombreux, les renards se multiplient. Plus il y a de renards, moins il y a de lièvres puisqu'ils sont mangés. Mais quand il y a moins de lièvres, le nombre de renards diminue : ils meurent de faim.

Ce que nous venons de décrire avec des mots doit, dans un premier temps, être transcrit en chiffres. Puis nous devons établir les formules correspondant aux règles que nous venons d'énoncer. Ainsi naît un modèle informatique de simulation du développement de la population des lièvres et des renards. On pourra ensuite, grâce à l'ordinateur, prévoir ce qui se passerait si l'on modifiait les données du modèle.

Données numériques

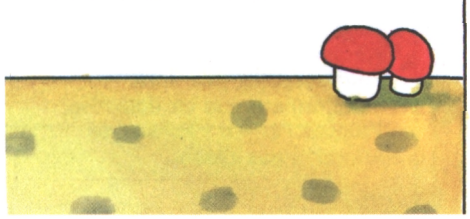
Supposons au départ que 'L' lièvres et 'R' renards vivent sur une île. A la fin d'une année, vivent 40 % de lièvres en plus. Mais parmi les 'L' lièvres, L*R/250 lièvres sont dévorés (250 est une sorte de ''facteur de gourmandise''). Si l'île est habitée de 'L' lièvres et de 'R' renards au début de l'année, elle sera peuplée l'année suivante par 'X' lièvres :

X = L + L*40/100 − L*R/250

Étudions maintenant le cas des renards. Leur taux de croissance est de (L/5−30)%. A la fin d'une année, le nombre total 'Y' de renards est :

Y = R + R*(L/5−30)/100

A partir de ces deux formules, on peut en déduire année par année le nombre de lièvres et de renards.



Les grandes lignes du programme

Le programme a besoin de connaître les données suivantes :

L : nombre de lièvres

R : nombre de renards

A : nombre d'années que dure l'étude

Ces trois valeurs seront introduites par l'instruction 'INPUT'.

Une boucle 'FOR ... NEXT' est parcourue 'A' fois. Les nombres de lièvres 'X' et de renards 'Y' seront calculés chaque fois en partant des données de l'année précédente : 'L' lièvres et 'R' renards. On utilise pour cela les deux formules citées précédemment. Mais il convient de ne garder que la partie entière des résultats (fonction 'INT'), car découper des animaux n'a ici aucune signification.

Le programme imprime ensuite les résultats. Il est possible que 'X' ou 'Y' devienne négatif avant que la boucle 'FOR ... NEXT' soit terminée. Il convient donc de vérifier à chaque boucle que 'X' et 'Y' sont positifs.

A la fin de chaque boucle, les nouvelles valeurs 'X' et 'Y' qui viennent d'être calculées deviennent les valeurs de départ pour l'année suivante. 'L' reçoit donc la valeur de 'X' et 'R' celle de 'Y'.

Commentaires

Les lignes 20 à 50 servent à introduire dans le programme, par le clavier, les valeurs initiales :
Nombre de lièvres 'L', nombre de renards 'R', et nombre d'années 'A'.
La boucle de calcul pour une année se situe entre 100 et 300. Cette boucle s'exécute 'A' fois. Le modèle mathématique est programmé aux lignes 110 et 120.
A la ligne 150, on vérifie qu'il reste des lièvres. Si oui, le programme continue à la ligne 160 ; si non, on termine la boucle 'FOR ... NEXT' en forçant le compteur 'N' à prendre immédiatement la valeur finale 'A' (ligne 155), et l'exécution passe à la ligne 200 : transferts entre variables pour préparer l'impression, impression, puis le programme s'arrête car la boucle est terminée et il ne reste plus d'instructions au-delà de la ligne 300.
Les lignes 160 et 166 jouent un rôle similaire pour les renards.
Notez que l'instruction END, n'étant pas ici indispensable, n'a pas été utilisée. En toute rigueur, il aurait fallu écrire :

310 END

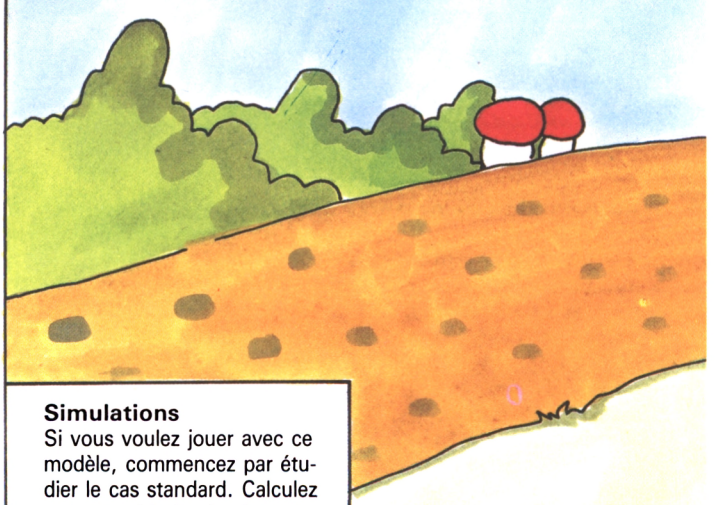
Le programme

```
10 REM RENARDS ET LIEVRES
20 INPUT "LIEVRES ";L
25 PRINT "LIEVRES : ";L
30 INPUT "RENARDS ";R
35 PRINT "RENARDS : ";R
40 INPUT "NOMBRE D'ANNEES ";A
45 PRINT "NOMBRE D'ANNEES : ";A
50 PRINT
100 FOR N=1 TO A
110 X=INT(L+L*40/100-L*R/250)
120 Y=INT(R+R*(L/5-30)/100)
150 IF X>0 THEN GOTO 160
155 N=A
156 GOTO 200
160 IF Y>0 THEN GOTO 200
165 N=A
166 GOTO 200
200 L=X
210 R=Y
250 PRINT N;" ";L;" ";R
300 NEXT N
```

Test du programme

Lorsque vous avez tapé votre programme, vérifiez d'abord qu'il ne comporte pas de faute de frappe. L'instruction 'LIST' vous permettra de le faire défiler à l'écran. Lisez-le attentivement et corrigez-le le cas échéant.
Pour vérifier que le programme calcule correctement, proposez-lui des données dont vous connaissez déjà le résultat. Si vous tapez les valeurs 100 pour les lièvres et 80 pour les renards sur 5 années, vous devrez obtenir les résultats suivants :

années
1re : 108 lièvres - 72 renards
2e : 120 lièvres - 65 renards
3e : 136 lièvres - 61 renards
4e : 157 lièvres - 59 renards
5e : 182 lièvres - 56 renards



Simulations

Si vous voulez jouer avec ce modèle, commencez par étudier le cas standard. Calculez sur une période plus longue (A = 20) avec 100 lièvres et 100 renards au départ. Que se passe-t-il ? Les lièvres se multiplient tandis que le nombre de renards diminue modérément. Que se passe-t-il si les lièvres se multiplient seulement avec un taux de croissance de 30 % tandis que les renards deviennent gloutons ? Pour cela, vous devez changer, à la ligne 110, la valeur 40 en 30 et la valeur 250 en 150. Puis relancez le programme avec à nouveau 'L' = 100 et 'R' = 100.
Vous constaterez qu'il est difficile de trouver les valeurs pour lesquelles les populations des lièvres et des renards se stabilisent. Bien sûr, il est également intéressant de définir les conditions de disparition des lièvres ou des renards. Sans l'ordinateur, la réponse à ces questions serait très longue à donner, voire même impossible. Voici pourquoi la simulation sur ordinateur est devenue une technique importante dans de nombreux domaines scientifiques.

Extensions et développements

Il est naturellement possible d'améliorer grandement notre programme. Cela dépend entre autres du type de matériel dont vous disposez. Ici, le programme est conçu pour fonctionner sur les plus petits ordinateurs. Les dialogues entre l'utilisateur et l'ordinateur restent notamment à un très bas niveau qu'il serait facile d'améliorer.

Le calendrier

En France, il est normal d'écrire une date sous la forme jour-mois-année. En Angleterre et aux Etats-Unis, l'ordre est différent : mois-jour-année.
On peut avoir besoin de connaître le nom du jour dans l'année correspondant à une date précise. Par exemple, le 14 juillet 1789 était-il un mardi ? Notre programme est capable de faire ce calcul ainsi que le calcul réciproque. Toutefois, nous ne pourrons remonter le temps au-delà de 1582, date à laquelle nous avons changé de calendrier pour adopter celui défini par le Pape Grégoire : le calendrier grégorien.

Le 15 octobre 1582 était un vendredi. En partant de cette origine, l'ordinateur calcule le jour correspondant à une date, semaine par semaine. On peut donc obtenir le calendrier de n'importe quelle année.

Transformation de nombres en noms
Comment peut-on retrouver le nom d'un mois, connaissant son numéro entre 1 et 12 ? A la ligne 100 du programme, nous effectuons un petit calcul faisant correspondre à chaque mois une ligne du programme. Ainsi, pour le troisième mois de l'année, 'N' prend la valeur 130. A cette ligne, nous plaçons 'MARS' dans la variable 0$. Cette procédure utilise donc une ligne par nom de mois. Nous procédons de la même manière pour découvrir le nom du jour de la semaine.

Comment fonctionne le calendrier grégorien
Si l'on veut connaître le jour de la semaine correspondant à une date du calendrier grégorien, on doit procéder ainsi : tout d'abord, il faut calculer le nombre de jours total écoulé entre le premier jour du calendrier grégorien et la date donnée. Il existe pour cela une formule tenant compte des années bissextiles et des différentes longueurs de mois. On divise ce résultat par 7 pour obtenir le nombre de semaines complètes écoulées. Le reste détermine le jour de la semaine, sachant que le premier jour est le lundi et le septième jour le dimanche.

Petite histoire du calendrier
Notre calendrier actuel est basé sur le temps que met notre planète pour effectuer une révolution complète autour du soleil. Mais cette révolution ne s'effectue pas exactement en 365 jours. Il reste à peu près un quart de jour d'erreur par an. Pour cette raison, tous les quatre ans, il est nécessaire d'ajouter une journée : c'est l'année bissextile. Toutefois, cela ne suffit pas encore et tous les 400 ans une année qui devrait être bissextile ne l'est pas. Enfin, tous les 3 000 ans, une petite correction est encore nécessaire.

Au Moyen Âge, on utilisait encore le calendrier julien tel qu'il avait été déterminé par Jules César en 46 avant Jésus-Christ. A la Renaissance commença une longue discussion sur la nécessité de définir un nouveau calendrier, car le calendrier julien ne correspondait plus du tout à la réalité des saisons. En 1582, Grégoire XIII définit son nouveau calendrier, largement utilisé aujourd'hui dans le monde entier. Bien entendu, notre programme suit ce calendrier qui ne convient toutefois pas pour les dates antérieures à 1582.

Le programme

```
5 REM CALENDRIER
10 INPUT "JOUR ";J
20 INPUT "MOIS ";M
30 INPUT "ANNEE ";Y
40 A=Y
50 REM MOIS DE L'ANNEE
100 ON M GOTO 110,120,130,140,150,
    160,170,180,190,200,210,220
110 O$="JANVIER":GOTO 300
120 O$="FEVRIER":GOTO 300
130 O$="MARS":GOTO 300
140 O$="AVRIL":GOTO 300
150 O$="MAI":GOTO 300
160 O$="JUIN":GOTO 300
170 O$="JUILLET":GOTO 300
180 O$="AOUT":GOTO 300
190 O$="SEPTEMBRE":GOTO 300
200 O$="OCTOBRE":GOTO 300
210 O$="NOVEMBRE":GOTO 300
220 O$="DECEMBRE":GOTO 300
300 IF M>2 THEN 330
310 M=M+12
320 A=A-1
330 N=J+2*M+INT(.6*(M+1))+A+INT(A/4)
   -INT(A/100)+INT(A/400)+2
350 N=INT((N/7-INT(N/7))*7+.5)
360 ON N+1 GOTO 400,410,420,430,440,
    450,460
400 A$="SAMEDI":GOTO 500
410 A$="DIMANCHE":GOTO 500
420 A$="LUNDI":GOTO 500
430 A$="MARDI":GOTO 500
440 A$="MERCREDI":GOTO 500
450 A$="JEUDI":GOTO 500
460 A$="VENDREDI":GOTO 500
490 REM IMPRESSION
500 PRINT A$;" ";J;" ";O$;" ";Y
510 GOTO 10
```

Les grandes lignes du programme

Pour utiliser notre programme, vous devez exprimer la date (jour-mois-année) par des nombres. Le numéro du mois permet de trouver la ligne dans laquelle la variable 0$ recevra le nom du mois correspondant. Puis le programme trouve le numéro du jour dans la semaine, ce numéro étant compris entre 0 et 6 (c'est la variable Y). De la même manière que pour les mois, cette variable permet d'accéder à une ligne du programme (entre 400 et 460) où la variante A$ recevra le nom du jour correspondant. Puis le programme imprime le résultat : nom du jour dans la semaine, numéro du mois et année.

Commentaires

Les lignes 10 à 30 servent à la saisie de la date : vous tapez le numéro du jour, le numéro du mois et l'année complète. Par exemple :

6
10
1956

Aucun test de validité n'est exécuté par le programme ; vous pouvez ainsi apprendre que le 31 février 1983 était un vendredi !
La ligne 100 sert à calculer le numéro de la ligne qui attribuera à 0$ le nom du mois. Entre les lignes 300 et 350, la donnée est convertie en jour de la semaine. Le calcul est très précis et il est nécessaire d'arrondir le résultat à la ligne 330 : 365,25 jours dans l'année et 30,6 jours par mois.
A la ligne 350, nous découpons le nombre total de jours entre l'origine du calendrier et la date donnée en tranches de sept jours. Ce qui nous intéresse, c'est le solde, c'est-à-dire le nombre de jours restants lorsqu'on a soustrait le plus grand nombre de semaines entières possibles : c'est le résultat de N – INT(N/7)*7. Ce nombre est compris entre 0 et 6. Puis il est multiplié par 10 et ajouté à 400, simplement pour former un numéro de ligne compris entre 400 et 460.

Test

Afin d'être certain du bon fonctionnement de ce programme, vous devez le tester. Deux tests simples vous le permettent.
Tout d'abord, essayez le programme avec sept dates consécutives, afin de vérifier que les noms des jours sont correctement générés.
Dans le second test, donnez douze dates correspondant au premier jour de chaque mois. Vous vérifiez ainsi que les noms des mois sont produits correctement. En procédant de cette manière, vous vérifiez également que les noms des jours sont justes.
Si ces tests fonctionnent normalement, le programme est certainement juste. Vous pourrez le confirmer en utilisant des dates extraites d'un calendrier officiel.
En cas d'erreur, vous devrez alors travailler minutieusement sur le programme, le faire avancer ligne par ligne, et vérifier après chaque calcul la validité du résultat. N'hésitez pas à utiliser le mode 'TRACE' dont nous avons parlé au début de ce livre.

Améliorations

Vous pouvez affiner et améliorer la partie saisie des données, notamment en vérifiant la validité des dates (numéro du jour, numéro du mois, date postérieure à 1582). De plus, le 29 février n'existe que lorsque le numéro de l'année est divisible par 4 sans générer de reste dans la division. 'A' étant l'année, si l'équation suivante est vérifiée :

A – INT(A – 4)*4 = 0

il s'agit d'une année bissextile.

Variation

Une variante importante de notre programme consisterait à l'utiliser comme sous-programme dans un programme plus important, un agenda par exemple. Il suffit pour cela de supprimer les lignes de saisie et de placer dans les variables J, M, A les trois nombres représentant la date. Appelez ce sous-programme par l'instruction 'GOSUB' et remplacez l'instruction 'END' de la ligne 510 par 'RETURN'.
Vous pouvez ainsi imprimer votre propre calendrier ou laisser ce programme fonctionner constamment dans votre ordinateur (dans ce cas, il est intéressant, si votre ordinateur possède la fonction 'TIME', de l'utiliser pour obtenir l'heure et un changement de jour automatique).

Deviner des nombres tirés au hasard

Ce jeu se présente ainsi : l'ordinateur tire un nombre au hasard entre 0 et 99. Vous devez le deviner et, pour cela, vous disposez de huit chances au total. Le jeu est plus intéressant lorsque l'ordinateur délivre des commentaires dont l'effet est d'autant plus surprenant qu'ils sont imprévisibles. Le commentaire émis est donc tiré au hasard parmi les dix pré-enregistrés. Pour cela, l'ordinateur tire un chiffre entre 0 et 9. Si le 5 sort, c'est le cinquième commentaire qui apparaîtra.
Ces nombres tirés au hasard sont produits par un générateur de nombres aléatoires.

Dans notre jeu, il existe trois circonstances pouvant provoquer l'apparition d'un commentaire : lorsque vous devez faire un nouvel essai, lorsque vous avez gagné, et lorsque vous avez perdu.

Ebauche du programme de tirage au sort
Notre programme crée donc un nombre que nous appellerons 'Z'. Comme nous limitons le nombre d'essais à huit, nous avons besoin d'un compteur 'K'. Il commencera à 0, puis sera incrémenté de 1 à chaque tour. Il faudra ensuite vérifier que la valeur 8 n'est pas encore atteinte.
A chaque essai, un message tiré au hasard invitera le joueur à taper un nouveau nombre mémorisé dans la variable 'G'. On compare ensuite 'G' à 'Z'. Si 'G' égale 'Z', le joueur a trouvé.
Si 'G' est supérieur à 'Z', le programme indique que votre nombre est trop grand, et inversement si 'G' est inférieur à 'Z', il indiquera que votre nombre est trop petit. Si 'Z' n'est pas trouvé après huit tours, le joueur a perdu et l'ordinateur donnera un dernier commentaire à nouveau tiré au hasard parmi dix.

Tirage au sort
Le nombre à découvrir est produit par un générateur de nombres aléatoires qui n'est rien d'autre qu'un programme. Considérons un nombre compris entre 0 et 1 et formé par les 10 premières décimales de $\frac{Pi}{10}$. Si vous multipliez ce chiffre par lui-même à chaque tour, vous obtiendrez des nombres formés de suites de chiffres au hasard. En élevant un nombre compris entre 0 et 1 au carré, le résultat est de plus en plus petit et tend vers 0. Pour cette raison, on vérifie à chaque fois que le nombre que l'on utilise est supérieur à 0,4. Sinon, on lui ajoute 0,35.

Nombres aléatoires
Le nombre de départ Q = 0,3141592654 changera chaque fois qu'on le multipliera par lui-même. Lorsqu'on a besoin d'un seul chiffre, on choisit le cinquième après la virgule. Si le nombre tiré au hasard doit comporter deux chiffres, on extrait les quatrième et cinquième chiffres après la virgule. Si Q est inférieur à 0,4, on lui ajoute 0,35.

Q0	0.3141592654	
Q1	0.4411075298	10
Q2	0.1945758528	7
Q3	0.2965628595	6
Q4	0.4180435313	4
Q5	0.1747603941	6
Q6	0.2753734712	7
Q7	0.3910919785	9
Q8	0.5492173206	1
Q9	0.3016396652	3

Le choix du commentaire
En fonction de la valeur de N, l'exécution passe à une ligne comprise entre 200 et 290. C'est le rôle de la ligne 140. On retrouve le même procédé aux lignes 420 et 620.

Le programme

```
  5 REM TROUVER DES NOMBRES TIRES AU
    HASARD, AVEC DES COMMENTAIRES
    ALEATOIRES
 10 PRINT "TROUVEZ UN NOMBRE ENTRE 0
    ET 99"
 20 PRINT "VOUS AVEZ 8 CHANCES..."
 30 PRINT "DONNEZ D'ABORD UN CHIFFRE
    ENTRE 0 ET 0.3"
 40 INPUT T
 50 Q=0.3141592654
 60 Q=Q+T
 65 REM UN NOMBRE AU HASARD EST TIRE...
 70 IF (Q<=0.4) THEN Q=Q+0.35
 71 Q=Q*Q
 72 R=INT(Q*100000)
 73 S=INT(Q*1000)*100
 74 Z=R-S
 95 REM LE JEU COMMENCE
100 K=0
110 IF K=8 THEN GOTO 600
120 K=K+1
130 GOSUB 900
140 ON N GOTO 200,210,220,230,240,
    250,260,270,280,290
195 REM DIX COMMENTAIRES CHOISIS
    AU HASARD
200 PRINT "COMMENTAIRE 1 : DEVINEZ
    MAINTENANT":GOTO 300
210 PRINT "COMMENTAIRE 2 ":GOTO 300
220 PRINT "COMMENTAIRE 3 ":GOTO 300
230 PRINT "COMMENTAIRE 4 ":GOTO 300
240 PRINT "COMMENTAIRE 5 ":GOTO 300
250 PRINT "COMMENTAIRE 6 ":GOTO 300
260 PRINT "COMMENTAIRE 7 ":GOTO 300
270 PRINT "COMMENTAIRE 8 ":GOTO 300
280 PRINT "COMMENTAIRE 9 ":GOTO 300
290 PRINT "COMMENTAIRE 10 ":GOTO 300
300 INPUT G
310 IF (G=Z) THEN 400
320 IF (G>Z) THEN 350
330 PRINT G;" EST TROP PETIT."
340 GOTO 110
350 PRINT G;" EST TROP GRAND."
360 GOTO 110
400 REM COMMENTAIRE SI ON A TROUVE
410 GOSUB 900
420 ON N GOTO 440,450,460,470,480,
    490,500,510,520,530,540,550
440 PRINT "COMMENTAIRE 11 : TROUVE
    ...?":GOTO 550
450 PRINT "COMMENTAIRE 12":GOTO 550
460 PRINT "COMMENTAIRE 13":GOTO 550
470 PRINT "COMMENTAIRE 14":GOTO 550
480 PRINT "COMMENTAIRE 15":GOTO 550
490 PRINT "COMMENTAIRE 16":GOTO 550
500 PRINT "COMMENTAIRE 17":GOTO 550
510 PRINT "COMMENTAIRE 18":GOTO 550
520 PRINT "COMMENTAIRE 19":GOTO 550
530 PRINT "COMMENTAIRE 20":GOTO 550
550 PRINT "VOUS AVEZ TROUVE EN ";K;"
    COUPS !"
560 GOTO 800
600 REM COMMENTAIRE EN CAS D'ECHEC
610 GOSUB 900
620 ON N GOTO 640,650,660,670,680,
    690,700,710,720,730,740,750
640 PRINT "COMMENTAIRE 21 : NON
    TROUVE":GOTO 750
650 PRINT "COMMENTAIRE 22 ":GOTO 750
660 PRINT "COMMENTAIRE 23 ":GOTO 750
670 PRINT "COMMENTAIRE 24 ":GOTO 750
680 PRINT "COMMENTAIRE 25 ":GOTO 750
690 PRINT "COMMENTAIRE 26 ":GOTO 750
700 PRINT "COMMENTAIRE 27 ":GOTO 750
710 PRINT "COMMENTAIRE 28 ":GOTO 750
720 PRINT "COMMENTAIRE 29 ":GOTO 750
730 PRINT "COMMENTAIRE 30 ":GOTO 750
750 PRINT "MON NOMBRE ETAIT ";Z;
    " BIEN SUR..."
760 GOTO 800
800 REM ON RECOMMENCE ?
810 PRINT "ON REJOUE (O/N) ?"
820 INPUT B$
830 IF B$="N" THEN GOTO 850
840 GOTO 70
850 PRINT "VOUS N'AVEZ PAS CONFIANCE
    EN VOUS, EH ?"
860 END
895 REM GENERATEUR DE NOMBRES ALEA
    TOIRES
900 IF Q<=0.4 THEN Q=Q+0.35
910 Q=Q*Q
920 R=INT(Q*100000)
930 S=INT(Q*10000)*10
940 N=R-S
950 RETURN
```

Commentaires tirés au hasard

Les lignes 5 à 74 servent à initialiser le générateur de nombres aléatoires. Le nombre que vous devez découvrir est placé à la ligne 70 dans la variable 'Z'. Le compteur de boucles est initialisé à la ligne 100 (K = 0), la boucle elle-même commençant à la ligne 110. Tout d'abord, il faut vérifier que le nombre maximum d'erreurs autorisées (8) n'a pas été atteint. Si ce n'est pas le cas, le compteur de boucles est incrémenté de 1 (ligne 120). Puis un chiffre est tiré au hasard et provoque l'affichage d'un commentaire parmi dix (lignes 130 à 290).

A la ligne 300, vous tapez votre premier essai. Si vous avez vu juste, le programme passe à la ligne 400 et un commentaire choisi au hasard s'affichera. Sinon, le programme vous indique si votre nombre est trop grand ou trop petit, et la boucle recommence à son début.

Le générateur de nombres aléatoires se situe entre les lignes 900 et 950. Il utilise les mêmes variables 'Q', 'R', 'S' et 'N' dont le premier générateur s'est servi au début du programme pour créer 'Z' ; cela engendre un hasard supplémentaire.

Améliorations

On peut étendre et compléter le programme ci-dessus de plusieurs manières. Si vous souhaitez augmenter le nombre de commentaires, vous devez utiliser des nombres aléatoires à deux chiffres ou plus. Il faut donc modifier la ligne 930 en copiant la ligne 73. Dans ce cas, vous pouvez définir 100 messages différents. Si ce nombre est trop grand, vous adapterez la ligne 940 afin de restreindre l'intervalle du nombre final. Par exemple, vous pouvez écrire $N = INT((R - S)/4)$. Veillez à utiliser une formule rendant tous les nombres équi-probables. (Si vous vous limitez à 25, tous les nombres compris entre 0 et 24 doivent sortir autant de fois sur un très grand nombre de tirages.)

Le problème du test

La construction de nos programmes étant relativement simple, les tests s'effectuent rapidement, du moins tant que le programme n'utilise pas un générateur de nombres aléatoires. En effet, dans ce dernier cas, on ne peut pas prévoir les résultats, et les tests systématiques deviennent très longs. Une meilleure méthode consiste à utiliser l'instruction TRACE et à faire fonctionner le programme pas à pas. Vérifiez "à la main" les calculs effectués par l'ordinateur jusqu'à ce que vous soyez sûr que les différents chemins utilisés dans les huit essais fonctionnent correctement.

Générateur de nombres aléatoires

Le BASIC comporte souvent une instruction ('RANDOM' ou 'RND') remplaçant les lignes 900 à 950. Si votre BASIC ne connaît pas cette instruction, recopiez les lignes 900 à 950 comme indiqué sur le listing ci-contre. La ligne 900 vérifie que le nombre n'est pas nul, ce qui arriverait en multipliant indéfiniment 'Q' par lui-même. La ligne 920 place dans 'R' les cinq premiers chiffres de 'Q' et la ligne 930 place dans 'S' les quatre premiers chiffres et leur ajoute un 0. Supposons : Q = 0.4246342533
'R' vaut alors 42463 et 'S' 42460.
En conséquence, N vaut :
N = 42463 − 42460 = 3
Ce résultat correspond bien à la cinquième décimale de 'Q'. C'est notre chiffre aléatoire. Bien sûr, si l'on commence toujours avec la même valeur pour 'Q', le générateur délivrera toujours la même suite de chiffres. Pour éviter cela, il est conseillé à l'utilisateur de définir lui-même un nombre entre 0 et 0,3.

Test

Voici quelques données avec leurs résultats qui vous faciliteront le test du programme. On pose : J = 0 (ligne 40).
'H' vaut 10.

Tour K	Hasard N	Message avant chaque saisie	Votre valeur G	Réponse G?H
1	7	MSG 08	50	plus grand
2	6	MSG 07	25	plus grand
3	4	MSG 05	12	plus grand
4	6	MSG 07	6	plus petit
5	7	MSG 08	9	plus petit
6	9	MSG 10	10	égalité
Gagné	1	MSG 12		Nouveau jeu

Répertoire téléphonique

Vous utilisez certainement un répertoire pour vos numéros de téléphone.
L'ordinateur est particulièrement apte à stocker des informations. Tenir à jour un répertoire téléphonique doit donc lui être très facile. Sur ces deux pages, vous trouverez sept petits programmes composant ensemble un répertoire téléphonique. Nous avons volontairement réduit les programmes afin qu'ils puissent être exécutés sur une calculatrice programmable en BASIC, mais l'ensemble ne peut gérer plus de dix noms et numéros de téléphone.
Ce programme complet est performant car, aux fonctions habituelles de recherche du nom ou du numéro de téléphone, s'ajoutent les fonctions de modification, d'effacement et d'ajout. Il est également possible d'imprimer le contenu du répertoire sur papier si vous possédez une imprimante.

L'ensemble des programmes

Notre programme travaille avec un tableau de données 'A$' dimensionné à 20. Nous y stockons les dix noms ainsi que les dix numéros de téléphone. Le contenu d'une case ne peut excéder sept caractères (sur les calculatrices programmables ; ce n'est pas vrai sur les micro-ordinateurs), ce qui oblige à ne jamais dépasser cette limite.
Les noms et les nombres sont enregistrés respectivement de A$(1) à A$(10) et de A$(11) à A$(20). Au début du programme, toutes les cases sont initialisées par des 'Z' à l'emplacement des noms, et par des '9' à l'emplacement des nombres. Cela permettra de savoir avec certitude si une case est vide ou non.
Voici la liste des programmes :

I — Menu
II — Recherche du numéro de téléphone correspondant à un nom donné
III — Recherche d'un nom correspondant à un numéro de téléphone donné
IV — Enregistrement d'un nouveau nom
V — Suppression d'un enregistrement existant
VI — Impression du répertoire
VII — Remise à zéro. Tout le tableau est initialisé avec ZZZZZZZ et 9999999.

Technique de recherche

L'algorithme de recherche utilisé dans nos programmes est très simple. Il consiste à comparer chaque élément du tableau avec l'élément recherché X$ jusqu'à détecter l'égalité. Appliquée à une recherche parmi les dix éléments, cette technique nécessite en moyenne cinq essais. En BASIC, nous utiliserons pour cela une boucle 'FOR ... NEXT'. Si l'on veut connaître l'élément dans le tableau A$(i) correspondant à X$, il faut écrire :

```
120 FOR Z = 1 TO 10
130 IF X$ = A$(Z) THEN 180
140 NEXT Z
150 PRINT "NOM"; X$; "INCONNU"
170 STOP
180 PRINT A$(Z)
```

Le compteur de boucles 'Z' indique en même temps la position de l'élément testé dans le tableau. Les éléments du tableau n'ont pas à être classés, ce qui simplifie l'algorithme. Sur des tableaux plus importants, le fait de classer des éléments par ordre alphabétique aurait permis d'utiliser d'autres algorithmes plus rapides.

Dimensions du tableau

Dans notre cas, le tableau a toujours la même dimension. On peut toutefois effectuer une recherche dans un tableau dont la dimension n'est pas constante. Il suffit alors de convenir que la recherche s'arrêtera lorsque l'on détectera la fin du tableau. Pour marquer cette fin, il suffit de placer dans la dernière case un mot qui ne risque jamais d'apparaître, par exemple 'AAAAAAA'. Notre programme de recherche doit alors fonctionner différemment en testant à chaque fois le contenu de la case pour savoir s'il s'agit de la dernière ou non. Nous n'utilisons plus de boucle 'FOR ... NEXT' mais nous devons utiliser un index 'Z' initialisé à 1 et que nous incrémentons de 1 après chaque test.

```
900 Z = 1
910 IF A$(Z) = "AAAAAAA" THEN
    950:REM FIN?
920 IF A$(Z) = X$ THEN 970:REM
    TROUVE?
930 Z = Z + 1
940 GOTO 910
950 PRINT "INEXISTANT!":REM FIN
960 STOP
970 PRINT A$(Z):REM TROUVE!
```

Programme I

```
1 REM MENU
5 CLEAR 1000
6 DIM A$(30)
10 PRINT:PRINT "1. RECHERCHE DU NUMERO PAR LE NOM"
20 PRINT "2. RECHERCHE DU NOM PAR LE NUMERO"
30 PRINT "3. NOUVELLE DONNEE"
40 PRINT "4. EFFACEMENT D'UN NUMERO"
50 PRINT "5. REPERTOIRE TELEPHONIQUE"
60 PRINT "6. REMISE A ZERO"
70 PRINT:INPUT"VOTRE CHOIX ";A
80 ON A GOTO 100,200,300,400,500,600
```

Programme II

```
99 REM RECHERCHE DE NUMERO PAR NOM
100 INPUT "TAPEZ LE NOM S.V.P.";X$
110 PRINT "LE NOM EST ";X$
120 FOR Z=1 TO 10
130 IF X$=A$(Z) THEN 180
140 NEXT Z
150 PRINT "NOM ";X$;" INCONNU."
160 PRINT "RECOMMENCEZ."
170 GOTO 10
180 Z=Z+10
181 PRINT "LE NUMERO EST ";A$(Z)
190 GOTO 10
```

Programme III

```
199 REM RECHERCHE D'UNNOM PAR UN NUMERO
200 INPUT "TAPEZ LE NUMERO ";Y$
210 PRINT "LE NUMERO EST ";Y$
220 FOR Z=1 TO 10
230 W=Z+10
235 IF Y$=A$(W) THEN 280
240 NEXT Z
250 PRINT "NOMBRE ";Y$;" INCONNU."
260 PRINT "RECOMMENCEZ."
270 GOTO 10
280 PRINT "LE NOM EST ";A$(Z)
290 GOTO 10
```

Programme IV

```
299 REM NOUVELLE ENTREE
300 PRINT "ESPERONS QU'IL RESTE DE LA PLACE..."
310 FOR Z=1 TO 10
320 IF A$(Z)="ZZZZZZZ" THEN 350
330 NEXT Z
340 PRINT "LE REPERTOIRE EST PLEIN..."
345 GOTO 10
350 INPUT "TAPEZ VOTRE NOUVEAU NOM ";A$(Z)
355 Z=Z+10
360 INPUT "TAPEZ LE NUMERO ";A$(Z)
370 GOTO 10
```

Programme V

```
399 REM EFFACEMENT D'UN ENREGISTREMENT
400 PRINT "ATTENTION ! EFFACEMENT"
410 INPUT "TAPEZ LE NOM A EFFACER ";X$
420 FOR Z=1 TO 10
430 IF A$(Z)=X$ THEN 460
440 NEXT Z
450 PRINT "NOM ";X$;" INCONNU."
455 GOTO 10
460 W=Z+10
465 PRINT A$(Z);" ";A$(W);" EFFACE."
470 A$(Z)="ZZZZZZZ"
480 A$(W)="9999999"
490 GOTO 10
```

Programme VI

```
499 REM LISTE DU REPERTOIRE
500 PRINT "LISTE DU REPERTOIRE"
510 FOR Z=1 TO 10
515 IF A$(Z)="ZZZZZZZ" THEN 530
520 PRINT A$(Z)
521 W=Z+10
522 PRINT A$(W)
530 NEXT Z
540 PRINT "FIN."
550 GOTO 10
```

Programme VII

```
599 REM REMISE A ZERO
600 PRINT "EFFACEMENT DE
    TOUTE LA LISTE"
610 FOR Z=1 TO 10
620 A$(Z)="ZZZZZZZ"
630 A$(Z+10)="9999999"
640 NEXT Z
645 PRINT "C'EST EFFACE."
650 GOTO 10
```

Programme I — Menu

Les instructions placées entre les lignes 5 et 70 affichent à l'écran les différentes possibilités offertes par l'ensemble des programmes. Ce programme utilise l'instruction 'ON A GOTO' suivie d'un numéro de ligne, pour lancer le programme voulu en fonction du choix de l'utilisateur.

Programme II — Recherche d'un numéro

Le programme vous demande de taper le nom dont vous souhaitez trouver le téléphone (ligne 100). A la ligne 110, l'ordinateur l'affiche à nouveau dans un but de confirmation. Les lignes 120 à 140 constituent la boucle de recherche qui scrute les cases A$(1) à A$(10) jusqu'à trouver une égalité. Si le nom est effectivement trouvé, le numéro est affiché (ligne 180-181). S'il n'existe pas dans le répertoire, un message vous en informe (ligne 150) et le programme revient au menu.

Programme III — Recherche d'un nom

Il fonctionne d'une manière similaire au programme précédent. Vous devez donner un numéro de téléphone qui s'appellera 'Y$'. Une recherche a lieu pour déterminer si ce numéro se trouve dans les cases A$(11) à A$(20). Si le numéro est effectivement trouvé, le programme affiche le nom correspondant ; sinon, un commentaire vous indique que ce numéro est inconnu. Le nom est trouvé grâce au compteur de boucles 'Z' servant d'indice dans le parcours du tableau A$(n).

Programme IV — Nouvelle entrée

Entre les lignes 300 et 345, le programme cherche la première case libre. Il recherche pour cela une case dont le contenu serait 'ZZZZZZZ'. Si le tableau est déjà plein, nous en sommes informés par la ligne 340. Si tout va bien, le programme demande ensuite le nom (ligne 350) et le numéro correspondant (ligne 360). Le nom se met dans la case A$(Z), le numéro se rangeant dans la case A$(Z + 10).

Programme V — Effacement d'un numéro

En partant du nom, le programme cherche la case correspondante pour y écrire 'ZZZZZZZ' puis '9999999' dix cases plus loin.

Programme VI — Liste du répertoire

Ce programme imprime la liste complète des noms et des numéros de téléphone. La ligne 515 détecte si la case est vide ou non : seules les cases effectivement utilisées sont imprimées.

Programme VII — Effacement de la liste

Cette opération n'est nécessaire que lorsque vous créez le tableau en mémoire pour la première fois. Utilisez-le donc avant de rentrer quoi que ce soit, cela vous assurera que les autres parties fonctionneront correctement sans produire d'absurdités. Toutes les cases A$(1) à A$(10) recevront le mot 'ZZZZZZZ' et toutes les cases A$(11) à A$(20), '9999999'.

Le test ne constitue pas un problème

Lorsque tous les programmes auront été tapés, commencez par utiliser le programme VII, et assurez-vous que toutes les variables A$(1) à A$(10) sont bien occupées par 'ZZZZZZZ'. Ensuite, passez au programme VI (impression de la liste). S'il fonctionne correctement, seul le titre s'affichera.
Utilisez ensuite le programme IV (nouvelle saisie) pour rentrer un nom et un numéro de téléphone dans le fichier. Reprenez ensuite le programme VI pour vérifier que vous obtenez bien la même chose que ce que vous avez tapé.
Si votre ordinateur dispose d'une mémoire importante, vous pouvez alors considérablement étendre les possibilités de l'ensemble. Pour connaître le nombre maximum de caractères que vous pouvez placer dans une variable alphanumérique, consultez votre manuel. Vous pourrez certainement aussi augmenter le nombre de noms et de numéros de téléphone si vous pouvez aller au-delà de A$(30), [par exemple, A$(100) vous permettrait d'enregistrer 50 noms et 50 numéros de téléphone].
N'oubliez pas de modifier l'instruction DIM en conséquence.

Améliorons le programme

Remarquez que les numéros sont stockés sous forme de chaînes de caractères. Rien ne vous interdit donc de différencier l'indicatif grâce à des parenthèses ou des tirets, par exemple (16-1) 43-55-47-23. Il faut, dans ce cas, que le programme précise la manière d'écrire un numéro au clavier. Vous pouvez aussi mémoriser les noms des villes dans d'autres cases du tableau, par exemple A$(21) à A$(30). Ceci constitue un bon exercice pour apprendre à modifier un programme.

L'ordinateur devine des nombres

Aux pages 14 et 15, nous avons étudié un programme dont le but était de vous faire deviner des nombres produits au hasard par l'ordinateur. Nous avons vu comment il était possible de commenter de manière originale et inattendue les différentes étapes de ce jeu.
Il s'agit maintenant du problème inverse : l'ordinateur doit deviner en moins de dix questions un nombre que vous aurez déterminé entre 1 et 127. Il le devine au plus tard au septième coup. Naturellement, il suppose que vous serez fairplay en restant dans la limite des nombres donnés. Sinon, il s'en rendra compte et vous le signalera.

L'utilisation de questions successives pour trouver un nombre implique quelque chose d'important : quel est le nombre minimum d'étapes nécessaires pour trouver un élément dans un tableau ? Nous étudierons ce point dans ce chapitre.

Principe de la recherche
Supposons que nous cherchions un nombre entre 1 et 3 ; nous demandons s'il ne s'agit pas de 2. Si la réponse est négative, nous demanderons si le nombre cherché est plus grand ou plus petit que 2. La réponse à cette question nous permettra de trouver la solution. Supposons que le nombre cherché soit compris entre 1 et 7. Raisonnons de la même manière. Le milieu de l'intervalle 1-7 est 4. Est-ce 4 ? Non ? Est-ce inférieur à 4 ? Si oui, le nombre est donc 1, 2 ou 3. Nous avons déjà étudié ce cas dans les lignes ci-dessus. Si le nombre est supérieur à 4, il vaut 5, 6 ou 7. Le problème est résolu en deux questions : 6 ? supérieur à 6 ? Cela semble compliqué, mais prenez une feuille de papier et écrivez les nombres de 1 à 15. Choisissez le nombre du milieu, c'est-à-dire 8. Puis déterminez le milieu de 1 et 8, c'est-à-dire 4. Continuez ainsi de suite jusqu'à déterminer le nombre maximum de questions nécessaires pour trouver un nombre compris dans cet intervalle. Vous constaterez qu'en quatre essais tout nombre est trouvé. Pour 31 nombres, cinq étapes sont nécessaires ; pour 63, six étapes.

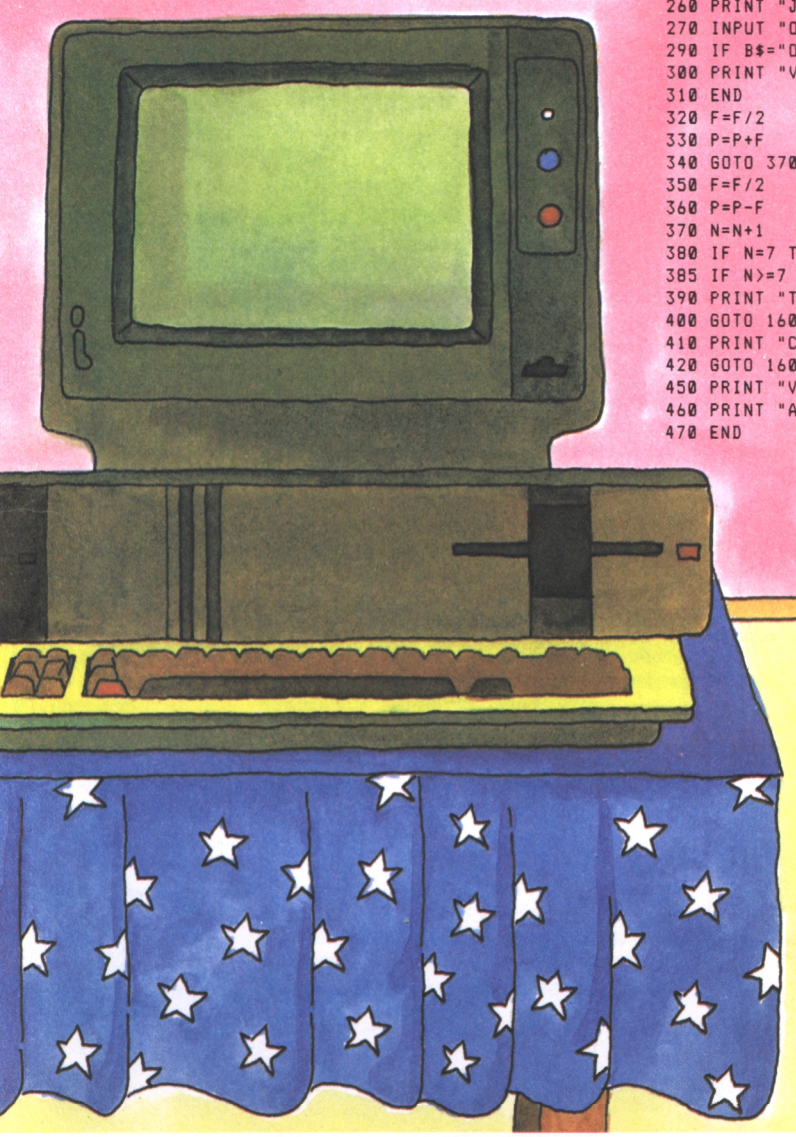
Trouvons un nombre

Quand on cherche un nombre entre 1 et 127, sept étapes sont nécessaires au maximum. 64 est le milieu de l'intervalle 1-127. Si le nombre recherché n'est pas 64, nous diviserons l'intervalle 1-127 en quatre parties (1 à 32, 33 à 64, 65 à 96, 97 à 127) et nous continuerons ainsi en divisant en deux chaque intervalle. Chaque fois que nous divisons par deux un intervalle, la quantité d'éléments que nous étudions est divisée par deux. Nous encadrons donc de plus en plus finement le nombre à trouver.

Notre programme prend 64 comme milieu entre 1 et 127. Si le nombre cherché est inférieur à 64, il divisera l'intervalle 1-64 à son tour en deux parties égales : 1-31 et 32-63. Si le nombre est supérieur à 64, le programme considère alors les intervalles 64-96 et 96-127 (c'est-à-dire 64 + 64/2 = 96). L'ordinateur applique cette technique jusqu'à trouver l'égalité. Il doit donc obligatoirement obtenir la réponse en sept essais maximum. S'il n'y arrive pas, vous n'avez pas choisi un nombre compris entre 1 et 127.

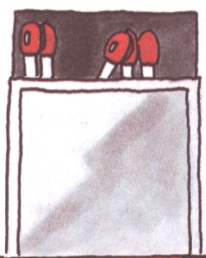
Le programme

```
5 REM L'ORDINATEUR DEVINE DES NOMBRES
100 PRINT "S'IL VOUS PLAIT, VEUILLEZ RESPECTER"
101 PRINT "LES REGLES ET NE CHANGEZ PAS DE NOMBRE"
102 PRINT "EN COURS DE JEU."
103 PRINT "JE DEVINE AU MAXIMUM EN 7 COUPS."
104 PRINT "LE NOMBRE QUE VOUS ALLEZ TAPER DOIT"
105 PRINT "ETRE COMPRIS ENTRE 1 ET 127."
106 PRINT "NE L'OUBLIEZ PAS !"
107 PRINT:PRINT "VOTRE REPONSE DOIT ETRE :"
108 PRINT "'T' POUR TROUVE!"
109 PRINT "'G' POUR TROP GRAND"
110 PRINT "'P' POUR TROP PETIT"
111 PRINT
150 P=64
151 N=1
152 F=64
160 PRINT "LE CHIFFRE EST PEUT-ETRE ";P;"..."
170 PRINT "QU'EN PENSEZ-VOUS ?"
180 INPUT A$
185 PRINT "VOUS AVEZ DIT ";A$;"..."
200 IF A$="T" THEN 260
210 IF A$="G" THEN 350
220 IF A$="P" THEN 320
230 PRINT "VOUS NE RESPECTEZ PAS LES REGLES !"
240 PRINT "TAPEZ T,G OU P !"
250 GOTO 180
260 PRINT "J'AI GAGNE EN ";N;" COUPS! LA CLASSE..."
270 INPUT "ON REJOUE ('OUI' OU 'NON') ";B$
290 IF B$="OUI" THEN 107
300 PRINT "VOUS MANQUEZ D'ENTRAIN. AU REVOIR !"
310 END
320 F=F/2
330 P=P+F
340 GOTO 370
350 F=F/2
360 P=P-F
370 N=N+1
380 IF N=7 THEN 410
385 IF N>=7 THEN 450
390 PRINT "TOUR";N
400 GOTO 160
410 PRINT "C'EST PRESQUE TROUVE..."
420 GOTO 160
450 PRINT "VOUS M'AVEZ TROMPE ! JE NE JOUE PLUS"
460 PRINT "AVEC VOUS !"
470 END
```



Le jeu de NIM

Le jeu de NIM est aussi vieux que simple. Pour y jouer, deux partenaires et une poignée d'allumettes suffisent. Au début de la partie, les joueurs se mettent d'accord sur le nombre d'allumettes utilisées. Les joueurs prennent tour à tour des allumettes, au moins une à chaque fois. Le nombre maximum que chacun peut prendre est déterminé au début de la partie. Celui qui doit prendre la dernière allumette a perdu. Notez qu'il existe d'autres versions de ce jeu utilisant plusieurs tas d'allumettes simultanément. Ce jeu est intéressant par la stratégie qu'il nécessite et qui fait perdre celui qui ne la connaît pas. Cette stratégie, parfaitement déterminée, peut être programmée, ce qui permet ensuite à n'importe qui de jouer contre l'ordinateur. Sur cette double page, nous décrivons le fonctionnement de ce programme.



Réalisation d'un jeu sur ordinateur

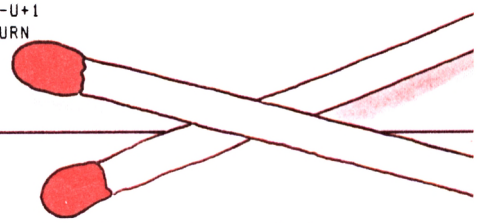
Pour réaliser un jeu sur ordinateur, il faut créer des conditions de jeu intéressantes, accompagnées de messages et de commentaires. Au début du jeu, l'ordinateur énonce clairement les règles et demande au joueur de lui fournir le nombre total d'allumettes ainsi que le nombre d'allumettes pouvant être prises à chaque coup. Le programme doit vérifier si ces nombres sont compatibles avec les règles. Ensuite, il faut déterminer qui commencera à jouer. Par correction, l'ordinateur laisse l'adversaire commencer. De toute manière, le joueur perd s'il ne connaît pas la stratégie. La troisième partie de notre programme est composée par le jeu lui-même. Entre les lignes 199 et 310, l'adversaire joue. Entre les lignes 319 et 480, c'est au tour de l'ordinateur. La stratégie de l'ordinateur est résumée en ligne 320. Elle n'est pas infaillible. Pour tromper son adversaire, l'ordinateur utilise un générateur de nombres aléatoires (ce programme est placé entre les lignes 800 et 850). Si l'adversaire est pris en flagrant délit de tricherie, le commentaire doit être mordant.

La stratégie pour gagner

Votre adversaire a perdu quand il trouve, à son avant-dernier tour, deux allumettes de plus que le maximum qu'il a le droit de prendre. S'il prend le maximum, il en reste donc deux. Vous n'en prenez qu'une et votre adversaire doit prendre la dernière. Il a donc perdu. S'il prend, à l'avant-dernier tour, une seule allumette, prenez alors le nombre maximum de manière à ce qu'il lui reste la dernière.
Vous connaissez le nombre total 'S' d'allumettes et le maximum 'M' auquel vous avez droit. Comme vous souhaitez que votre adversaire perde, vous calculez 'S − 1'. Décomposez ensuite ce nombre par paquets de 'M + 1' allumettes. Supposons que le nombre total d'allumettes soit 22, et que 'M' vaille 4. Vous obtenez : 'S − 1' = 21. Divisez ce nombre en paquets de 'M + 1' = 5 allumettes. Cela fait 4 paquets. Il reste une allumette isolée. Lorsque c'est à votre tour de jouer, prenez cette allumette. Puis prenez ensuite à chaque tour le complément à 5 ('M + 1') du nombre d'allumettes jouées par votre adversaire. Le mieux est d'utiliser cette stratégie dès le début du jeu si vous le pouvez. Si vous appliquez cette stratégie au deuxième ou au troisième tour et si votre partenaire la connaît et l'a déjà appliquée, vous ne pourrez rattraper votre handicap. S'il a pris les allumettes au hasard, vous avez encore toutes vos chances. Vous pourrez ensuite calculer en combien de temps vous allez gagner.

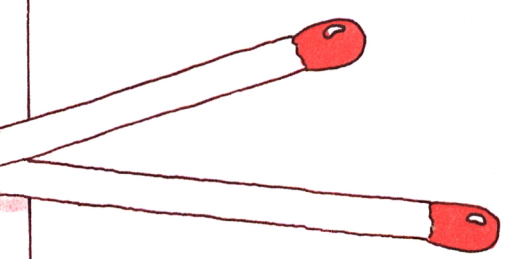
Le programme

```
10 REM LE JEU DE NIM
11 PRINT "LE GENERATEUR DE NOMBRES ALEATOIRES VA"
12 PRINT "SE PREPARER. TAPEZ UN NOMBRE ENTRE"
13 INPUT "0 ET 0.3 ";T
14 Q=0.3141592654 +T
19 REM REGLES DU JEU ET PREPARATION
20 PRINT "CHACUN, NOUS PRENDRONS AU MOINS UNE"
21 PRINT "ALLUMETTE A CHAQUE TOUR ;"
22 PRINT "ET AU PLUS, LE NOMBRE QUE VOUS ALLEZ"
23 PRINT "DEFINIR. CELUI QUI PRENDRA LA DERNIERE"
24 PRINT "ALLUMETTE AURA PERDU."
30 PRINT "VEUILLEZ MAINTENANT REPONDRE :"
31 PRINT "COMBIEN D'ALLUMETTES AU TOTAL ?"
32 INPUT S
33 PRINT S;" ALLUMETTES AU TOTAL..."
90 IF S>=20 THEN 120
100 PRINT "CE N'EST PAS BEAUCOUP. ENFIN!"
120 PRINT "COMBIEN PEUT-ON EN PRENDRE AU MAXIMUM ?"
130 INPUT M
131 PRINT "ON PEUT DONC EN PRENDRE ";M;" AU"
132 PRINT "MAXIMUM..."
140 IF M>=2 THEN 170
150 PRINT "CE CAS EST ININTERESSANT."
160 GOTO 120
170 IF S>M+1 THEN 200
180 PRINT "IL EN FAUDRAIT PLUS."
190 GOTO 30

199 REM TOUR DU JOUEUR
200 PRINT "A VOTRE TOUR."
210 INPUT "COMBIEN D'ALLUMETTES ";Z
220 IF Z>0 THEN 230
221 PRINT "VOUS DEVRIEZ AVOIR HONTE DE VOULOIR ME"
222 PRINT "TROMPER. DONNEZ UNE VALEUR DIFFERENTE"
223 PRINT "DE ZERO."
224 GOTO 200
230 IF Z<=M THEN 260
240 PRINT "VOYEZ-VOUS CE TRICHEUR!"
250 GOTO 200
260 S=S-Z
261 PRINT "VOUS AVEZ PRIS ";Z;" ALLUMETTES."
262 PRINT "IL EN RESTE DONC ";S
270 IF S>=1 THEN 320
280 IF S=0 THEN 460
290 PRINT "RESPECTEZ LES REGLES..."
300 S=S+Z
310 GOTO 200

319 REM TOUR DE L'ORDINATEUR
320 Z=S-1-INT((S-1)/(M+1))*(M+1)
330 IF Z<>0 THEN 370
340 IF M<=S THEN V=M:GOTO 342
341 V=S
342 GOSUB 800
343 IF N<=V THEN 346
344 N=N-V
345 GOTO 343
346 Z=N
350 PRINT "VOUS ME COMPLIQUEZ LA VIE."
360 GOTO 380
370 PRINT "VOUS NE VOULEZ VRAIMENT PAS GAGNER."
380 S=S-Z
390 PRINT "JE PRENDS ";Z;" ALLUMETTES."
391 PRINT "IL EN RESTE DONC ";S
440 IF S=0 THEN 480
450 GOTO 200
460 PRINT "VOUS AVEZ PERDU."
470 GOTO 490
480 PRINT "VOUS AVEZ ENFIN GAGNE."

489 REM NOUVEAU JEU ?
490 PRINT "ENCORE UN PETIT TOUR ?"
500 INPUT "('OUI' OU 'NON')";E$
510 IF E$="OUI" THEN 30
520 PRINT "AU REVOIR."
530 END

799 REM GENERATEUR DE NOMBRES ALEATOIRES
800 IF Q<=0.4 THEN Q=Q+0.35
810 Q=Q*Q
820 R=INT(Q*100000)
830 U=INT(Q*10000)*10
840 N=R-U+1
850 RETURN
```



Compter de manière stratégique

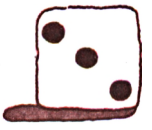
La stratégie du jeu est programmée entre les lignes 320 et 380. On divise 'S – 1' par 'M + 1' et on multiplie la partie entière du nombre par 'M + 1'. On soustrait le résultat de 'S – 1'. Il reste le nombre d'allumettes à prendre : 'Z'. Si 'Z' égale 0, on prend un nombre aléatoire d'allumettes compris entre 1 et 'M' ou, si 'S' est inférieur à 'M', compris entre 1 et 'S'. En outre, si le



programme passe à la ligne 370 et si aucune erreur n'est commise, la ligne 370 sera répétée à chaque tour.

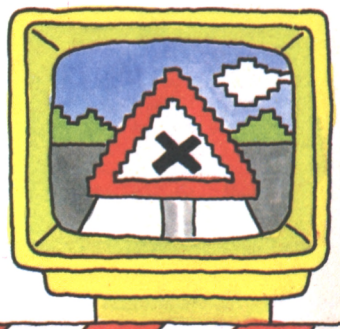
Commentaires

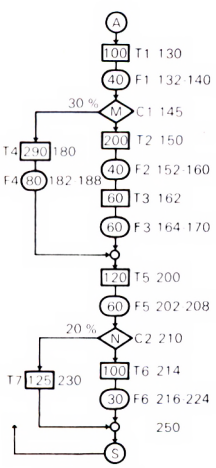
Le programme est d'une réalisation simple et facile à comprendre lorsque l'on connaît parfaitement la stratégie utilisée. Le générateur de nombres aléatoires est initialisé à la ligne 14. Les règles du jeu sont affichées par les lignes 19 à 24. Le programme demande ensuite le nombre maximum d'allumettes : 'S'. Ce nombre diminue à chaque tour puisqu'on enlève un certain nombre d'allumettes à chaque fois. 'M' est le nombre maximum d'allumettes que l'on peut retirer à chaque tour. On le tape à la ligne 130. Si 'M' est égal à 1 ou 'S' est inférieur ou égal à 'M + 1', le programme refuse ces données. Les données fournies par le joueur sont traitées par les lignes 199 à 310. S'il tape zéro allumette, la donnée sera refusée. Il en est de même s'il donne une valeur supérieure au maximum convenu. Si le partenaire a pris plus d'allumettes qu'il n'en restait, le programme affiche un message par la ligne 290 et le tour est annulé. A partir de la ligne 320, c'est au tour de l'ordinateur de jouer. Les lignes 490 à 512 servent à tester s'il est utile de poursuivre la partie. Enfin, entre 800 et 850, se situe le générateur de nombres aléatoires.



Simulation

La simulation, c'est-à-dire l'opération qui consiste à imiter des faits réels sur ordinateur, constitue une branche importante de l'informatique. Un exemple de ce genre est l'étude de la circulation routière. Vous pouvez suivre le schéma ci-dessous. On conduit d'un point 'A' vers un point 'S'. Les rectangles représentent les portions de routes à vitesse réglementée. Les numéros inscrits à l'intérieur représentent les secondes nécessaires pour parcourir le tronçon. Les petites boîtes ovales simulent les feux de circulation. Le nombre placé à l'intérieur représente le temps maximum d'attente en secondes. Toute attente entre 0 et ce maximum est possible. Les losanges représentent les carrefours. 'M' et 'N' représentent les pourcentages de conducteurs bifurquant au lieu de continuer tout droit. Chaque trajet à travers les rues est différent, comme d'ailleurs nous l'apprend la vie quotidienne. Les temps d'attente sont fixés par un générateur de nombres aléatoires. Les temps de conduite et d'attente sont additionnés pour que l'on puisse comptabiliser la durée totale de trajet. A chaque carrefour, un nombre aléatoire entre 1 et 100 est produit. S'il est inférieur à 'N' ou 'M', on bifurque ; sinon, on continue tout droit. Il est important d'étudier de nouveaux cas pour en déduire ensuite une règle fiable.

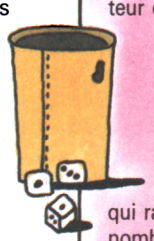




T : Tronçon
F : Feux tricolores
C : Croisement

La simulation des événements composant la circulation routière

Notre exemple comprend deux carrefours, sept portions de route à vitesse non réglementée, c'est-à-dire dont les durées de trajet sont comprises entre 60 et 290 secondes, et six feux tricolores avec un temps d'attente compris entre 30 et 80 secondes maximum. Les croisements sont très faciles à utiliser. Chacun est caractérisé par le pourcentage d'automobilistes empruntant la bifurcation. Chaque chemin individuel est une simple combinaison de temps de parcours et de période d'attente. Le plus court chemin dure 580 secondes, le plus long 810. Le hasard est créé par un générateur de nombres aléatoires produisant un nombre entier de deux chiffres compris entre 00 et 99. Si ce nombre doit être plus petit que 40, on s'arrange pour lui soustraire 40 autant de fois que nécessaire. En dessous de 20, le générateur de nombres n'est plus très performant car les nombres tirés au hasard entre 80 et 100 produisent un déséquilibre en raison de cette soustraction de 40. Si vous désirez améliorer ce générateur, arrangez-vous pour que tous les nombres supérieurs à 80 soient rejetés. Cela prend du temps. Comme nous ne nous intéressons ici qu'au principe de la simulation, nous conserverons ce déséquilibre.



Le programme de simulation

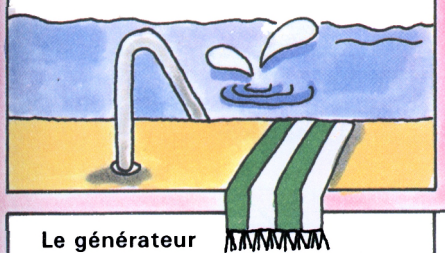
Notre programme n'est pas compliqué. Il demande les valeurs des nombres 'M' et 'N' représentant les pourcentages des usagers bifurquant aux carrefours. Il demande également le nombre de passages qu'il doit simuler ainsi que le nombre de départ pour le générateur de nombres aléatoires (sinon, la suite de nombres générés serait toujours la même).

Une boucle utilisant 'K' comme compteur est parcourue 'L' fois (lignes 106 à 252). Sur le dessin ci-contre sont indiqués les numéros de ligne correspondant à chaque phase.

Une simulation dépend des valeurs des nombres utilisés. Ces nombres sont appelés, en langage technique, "paramètres". De nombreux paramètres sont utilisés dans ce programme. Ainsi, 'N', 'M', 'L' et le pondérateur de nombres aléatoires sont des paramètres. Ils seront donc imprimés chaque fois, en même temps que la durée du trajet. Il devient ainsi facile de comparer différents cas et, par exemple, de savoir si l'on roule mieux en prenant le chemin le plus court mais en rencontrant des feux, ou en bifurquant, ce qui rallonge le parcours mais diminue le nombre de feux rencontrés.

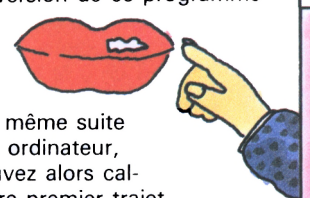
L'heuristique ou l'approche empirique

Archimède avait découvert, aux environs de 240 avant Jésus-Christ, son fameux principe d'hydrostatique : tout corps plongé dans un liquide est soumis par celui-ci à une force verticale, dirigée de bas en haut, équivalente au poids du volume de liquide déplacé. Il en fut si content qu'il courut tout nu dans la rue en criant "Heureka !", c'est-à-dire "J'ai trouvé !" (ce mot s'est ensuite traduit en français par "Euréka"). De ce verbe grec nous vient le mot "heuristique" désignant la discipline qui se propose de dégager les règles de la découverte (et non pas d'énoncer des théorèmes). L'intuition, l'empirisme, en font partie.
La simulation en est un exemple typique.
Comme le montre notre application, elle combine des paramètres contrôlables par l'homme avec la construction d'un modèle de référence.
Si l'on pose 'M' et 'N' égaux à 0, le modèle ne comporte plus de carrefour. Si l'on désire plus de deux carrefours, le modèle doit être reprogrammé.



Le générateur de nombres aléatoires

Lorsqu'on choisit, pour 'Q', une valeur de départ égale à 0.314259621, notre générateur fournit les dix premières valeurs suivantes (entre 0 et 99) :
10 57 56 4 76 37 9 21 64 63
Si votre version de ce programme



donne la même suite sur votre ordinateur, vous pouvez alors calculer votre premier trajet.
Pour 'M' = 30, 'N' = 20 et 'L' = 1, notre ordinateur a donné une durée de 635 secondes.

Simulation et durée du calcul

Les modèles de simulation sont assez longs à utiliser car les programmes sont eux-mêmes longs, et il faut réaliser beaucoup d'essais, en variant les valeurs des paramètres, pour obtenir un résultat fiable. C'est pourquoi les programmes de simulation fonctionnent généralement sur de gros ordinateurs.

Remarques

Comme beaucoup de programmes de simulation, notre programme comporte trois parties. Dans la première partie, les paramètres sont définis (lignes 10 à 54). Puis les compteurs sont remis à 0. La seconde partie du programme représente le modèle caractéristique des parcours (de la ligne 106 à la ligne 250). 'K' est le compteur de boucles, 'L' le nombre total de boucles à effectuer. 'T' totalise la durée de chaque parcours et est remis à 0 au début de chaque boucle. 'S' totalise les valeurs de 'T'. La troisième partie commence en 270. 'L', 'M', 'N' et 'X' sont affichés ainsi que la durée moyenne du parcours 'S/L'. Le sous-programme du générateur de nombres aléatoires va de la ligne 900 à la ligne 906.
Pourquoi n'avons-nous pas utilisé les instructions 'FOR ... NEXT' pour la boucle principale (lignes 106, 107, 252) ? Tout simplement parce que certaines petites machines de poche les ont refusées, certainement en raison du grand nombre de 'GOTO' et de 'GOSUB' utilisés. Pour vous éviter tout problème sur votre ordinateur, nous avons préféré nous passer de ces instructions.

Le modèle exact

Vous pouvez vous rapprocher d'un modèle exact en constatant que la durée du trajet varie d'une façon aléatoire sur les portions de routes dépourvues de feux tricolores. Pour cela, il vous suffit d'introduire des nombres aléatoires supplémentaires dans le programme.

Test

Les programmes de simulation sont difficiles à tester. D'une part, parce qu'ils fonctionnent souvent avec un générateur de nombres aléatoires, d'autre part parce que les calculs sont longs et nombreux. Le mieux est de faire exécuter le programme pas à pas. Testez d'abord le générateur de nombres aléatoires et notez les dix premières valeurs qu'il fournit.
Avec notre générateur, nous obtenons une durée de 635 secondes pour un seul parcours ('L' = 1), sans bifurquer aux carrefours ('M' = 0, 'N' = 0, 'X' = 0). Pour 'X' = 0, 'M' = 100, 'N' = 0 et 'L' = 1, cette durée passe à 687. Enfin, pour 'X' = 0, 'M' = 0, 'N' = 100 et 'L' = 1, nous obtenons 651.

Le programme

```
10 REM SIMULATION DE LA CIRCU
   LATION ROUTIERE
20 INPUT "CARREFOUR M ";M
25 PRINT "M=";M
30 INPUT "CARREFOUR N ";N
35 PRINT "N=";N
40 INPUT "NOMBRE DE TOURS ";L
45 PRINT "T=";L
50 PRINT "TAPEZ UN NOMBRE ENTRE
   0 ET .3"
51 INPUT "";X
54 Q=0.314159625 +X
100 S=0
105 K=0
106 K=K+1
107 IF K>L THEN 270
120 T=0
130 T=T+100
132 GOSUB 900
134 IF Z<=40 THEN GOTO 140
136 Z=Z-40
138 GOTO 134
140 T=T+Z
145 GOSUB 900
147 IF Z<M THEN GOTO 180
150 T=T+200
152 GOSUB 900
154 IF Z<=40 THEN GOTO 160
156 Z=Z-40
158 GOTO 154
160 T=T+Z
162 T=T+60
164 GOSUB 900
166 IF Z<=60 THEN GOTO 170
168 Z=Z-60
170 T=T+Z
172 GOTO 200
180 T=T+290
182 GOSUB 900
184 IF Z<=80 THEN GOTO 188
186 Z=Z-80
188 T=T+Z
200 T=T+120
202 GOSUB 900
204 IF Z<=60 THEN GOTO 208
206 Z=Z-60
208 T=T+Z
210 GOSUB 900
212 IF Z<N THEN GOTO 230
214 T=T+100
216 GOSUB 900
218 IF Z<=30 THEN GOTO 224
220 Z=Z-30
222 GOTO 218
224 T=T+Z
226 GOTO 250
230 T=T+125
232 GOTO 250
250 S=S+T
252 GOTO 106
270 PRINT "NOMBRE DE TESTS = ";L
280 PRINT "CARREFOUR M = ";M
290 PRINT "CARREFOUR N = ";N
295 PRINT "NOMBRE ALEATOIRE
    DE DEPART = ";X
300 U=S/L
310 PRINT "LA DUREE MOYENNE
    DU TRAJET EST DE ";U
312 PRINT "FIN DU TEST."
315 END
900 IF Q>=.4 THEN GOTO 902
901 Q=Q+.35
902 Q=Q*Q:PRINTQ;" ";
903 V=INT(Q*100000)
904 W=INT(Q*1000)*100
905 Z=V-W:PRINTZ
906 RETURN
```

Peut-on passer des nombres au crible ?

Cette question semble bizarre. Voici l'histoire d'un méchant roi qui avait jeté la moitié de ses sujets en prison et qui avait besoin de l'autre moitié pour leur servir de gardiens.
Avec le temps, il se rendit à l'évidence : on ne peut régner de cette façon ! Il chargea donc le mathématicien de la cour d'inventer une méthode selon laquelle quelques-uns de ses sujets seraient libérés.
Le mathématicien proposa la méthode suivante :
Chaque prisonnier a une cellule. Ces cellules sont numérotées dans l'ordre croissant. Le premier gardien ouvre toutes les portes. Le second s'intéresse seulement aux portes portant un numéro pair (c'est-à-dire à une porte sur deux) : si la porte est ouverte, il la referme ; si elle est fermée, il la rouvre. Le troisième gardien s'intéresse à une porte sur trois et applique la même règle, toujours en commençant au début. Et ainsi de suite. Quels seront les numéros des portes finalement ouvertes ?
Si l'on appelle cette méthode "Passer les nombres au crible", la question devient : "Quels seront finalement les nombres sélectionnés ?".

La sélection réalisée électroniquement par ordinateur

Si l'on veut adapter cette histoire sur ordinateur, il faut construire un modèle mathématique représentant la prison et le fonctionnement des portes. Le plus simple consiste à faire correspondre à chaque porte une variable recevant la valeur 0 si la porte est fermée et 1 si la porte est ouverte. En fait, nous utiliserons un tableau de nombres, c'est-à-dire un ensemble de variables ayant la même "racine" : seul l'indice change. Si nous choisissons 'A' comme racine, les éléments du tableau représentant les portes s'appelleront 'A(1)', 'A(2)', 'A(3)', etc., 1, 2 et 3 étant des indices. Nous créons dans la mémoire de l'ordinateur ce tableau après avoir tapé au clavier sa taille par l'intermédiaire de l'instruction 'INPUT'. Si ce nombre est 'B', notre prison a donc 'B' cellules et 'B' gardiens. Nous passons en revue ce tableau au moyen d'une boucle 'FOR ... NEXT' et nous changeons les 1 en 0 et les 0 en 1 selon le crible décrit. Lorsque l'opération est terminée, nous relisons le tableau par une autre boucle 'FOR ... NEXT'. Si la valeur est 0, la porte est fermée, nous l'ignorons et nous passons à la suivante. Si la valeur est 1, nous imprimons la valeur de l'indice : quels nombres vont alors apparaître ?

Un programme qui passe les nombres au crible

Si nous voulons essayer plusieurs cribles, nous diviserons le programme en trois parties. Tout d'abord, nous déterminerons le nombre de valeurs à passer au crible, c'est-à-dire la taille du tableau 'A'. Tous les éléments de ce tableau sont initialisés à zéro.
Dans la seconde partie, nous programmons le crible lui-même. Si nous désirons par la suite changer cette partie et tester d'autres algorithmes, seules ces lignes seront à modifier.
La troisième partie est réservée à l'affichage des résultats. Cela s'effectue dans une boucle passant en revue tous les éléments du tableau, de 1 à B. Si A(n) est égal à zéro, rien ne se passe ; si A(n) est égal à 1, 'n' est affiché.
Le programme calcule beaucoup car il doit souvent manipuler le tableau 'A'.

Le crible d'Eratosthène

Un Grec ancien du nom d'Eratosthène (276-194 avant J.-C.) découvrit un autre algorithme : toutes les portes sont ouvertes par le premier prisonnier sortant de sa cellule. Le deuxième laisse la sienne ouverte et ferme ensuite une porte sur deux. Et ainsi de suite : un prisonnier dont la porte est ouverte la laisse ouverte et ferme toutes les portes dont le numéro est un multiple de la sienne. On peut deviner ce qui arrive alors : seules restent ouvertes les portes dont le numéro n'est divisible que par 1, et par lui-même : ce sont les nombres premiers, et cet algorithme fut le premier permettant de les trouver.

Avec ou sans DIM ?

Nous avons testé ce programme sur un ordinateur de poche SHARP PC 1211. Il possède une imprimante et un BASIC intégré standard mais ne reconnaît pas L'INSTRUCTION DIM. On ne peut indicer que des variables dont le nom est 'A' et tant qu'il reste de la place en mémoire. En conséquence, notre tableau représentant les portes commence à 11, ce qui nous laisse une dizaine de cases libres ((A,0) à (A,10)) pour les autres variables dont notre programme a besoin. Voilà pourquoi nous voyons apparaître la variable A(10) à certaines lignes, comme par exemple en 260. Notez que le listing ci-contre fonctionne sans aucune modification sur les ordinateurs THOMSON ; pour l'ordinateur SHARP PC 1211, il faut supprimer l'instruction DIM.
Si votre ordinateur est plus performant que l'ordinateur de poche PC 1211 (c'est-à-dire s'il possède une plus grande mémoire vive), vous pouvez alors créer des tableaux de taille bien supérieure. Sur de nombreux ordinateurs, l'instruction 'DIM' est facultative tant que vous ne dépassez pas l'indice 10 ; au-delà, cette instruction est nécessaire pour indiquer à l'ordinateur la taille du tableau qu'il doit prévoir. Si vous oubliez 'DIM', l'ordinateur vous le rappellera en arrêtant l'exécution du programme et en affichant un message d'erreur.

Test pour le crible d'Eratosthène

Ce genre de programme s'apparente plus à la logique qu'à des calculs. Par conséquent, il est nécessaire de bien le tester. Le mieux consiste à choisir un tableau de taille raisonnable (10 par exemple) et d'examiner le programme pas à pas. Voici les données du test :

Numéro des cellules / Nombre de passages	1	2	3	4	5	6	7	8	9	10
1	1	1	1	1	1	1	1	1	1	1
2	1	1	1	0	1	0	1	0	1	0
3	1	1	1	0	1	0	1	0	0	0
4	1	1	1	0	1	0	1	0	0	0
5	1	1	1	0	1	0	1	0	0	0
6	1	1	1	0	1	0	1	0	0	0
7	1	1	1	0	1	0	1	0	0	0
8	1	1	1	0	1	0	1	0	0	0
9	1	1	1	0	1	0	1	0	0	0
10	1	1	1	0	1	0	1	0	0	0

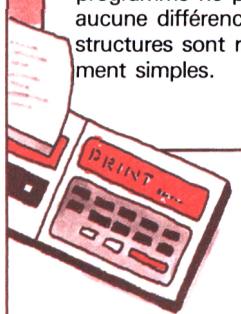
On constate que les portes ne changent plus d'état dès le quatrième passage. Comme le programme comporte beaucoup de calcul, cela vaut la peine d'en tenir compte : on peut arrêter le programme au nombre de passage suivant la valeur de la racine carrée du nombre total de cellules. Ici, le nombre total est 10, la racine est 3, on peut donc s'arrêter à 4.

Le programme

```
100 DIM A(120)
200 REM CRIBLE DES CARRES
210 PRINT "TAPEZ LE NOMBRE
    D'ELEMENTS"
220 INPUT "";B
230 PRINT "LE NOMBRE D'ELE
    MENTS EST ";B
250 FOR C=1 TO B
260 A(C+10)=0
270 NEXT C
300 FOR C=1 TO B
310 D=1
315 E=D*C+10
320 IF A(E)=0 THEN
    A(E)=1:GOTO 340
330 A(E)=0
340 D=D+1
350 IF ((D*C-1)>B) THEN 370
360 GOTO 315
370 NEXT C
600 FOR C=1 TO B
610 IF A(C+10)=0 THEN 630
620 PRINT C
630 NEXT C
640 PRINT "FIN"
650 END
```

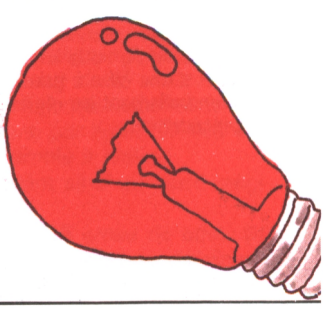
Le programme pour le crible d'Eratosthène

Jouons d'abord avec le crible du méchant roi. Il ne présente aucune difficulté. Il est ensuite facile de passer de ce crible à celui d'Eratosthène : il suffit d'initialiser à 1 toutes les cellules au départ, puis de commencer le crible avec la cellule numéro 2. A part cela, le programme ne présente aucune différence. Leurs structures sont relativement simples.



Le secret du roi

Le crible du méchant roi détermine tous les carrés parfaits entre 1 et la dernière cellule du tableau. Entre 1 et 10, nous devons donc obtenir 4 et 9. Quels carrés trouve-t-on entre 1 et 100 ? Votre calculatrice peut-elle deviner si 121 est un carré ?



pour les ordinateurs plus puissants

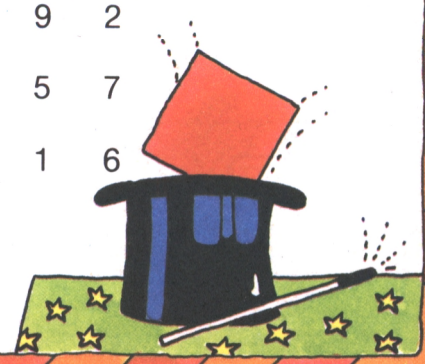
Nous avons sciemment réduit ce programme à sa plus simple expression, car nous voulions qu'il fonctionne sur une petite calculatrice de poche utilisant le BASIC (cela est d'ailleurs vrai pour tous les programmes contenus dans ce livre) ; les personnes possédant un plus gros ordinateur ont de la chance ! Elles pourront chercher à travailler sur des valeurs plus grandes et répondre aux questions suivantes : '' 'X' est-il un nombre premier ?'', ou ''Quel est le prochain nombre premier après 'Y' ?'', ou ''Combien de nombres premiers puis-je trouver avec ma calculatrice ?'', etc.

```
100 DIM A(200)
200 REM CRIBLE DES NOMBRES
    PREMIERS
210 PRINT "TAPEZ LE NOMBRE
    D'ELEMENTS"
220 INPUT "";B
230 PRINT "LE NOMBRE D'ELE
    MENTS EST ";B
250 FOR C=1 TO B
260 A(C+10)=1
270 NEXT C
300 FOR C=2 TO B
310 IF A(C+10)=0 THEN 370
315 E=D*C+10
320 D=2
330 A(C*D+10)=0
340 D=D+1
350 IF ((D*C-1)>B) THEN 370
360 GOTO 330
370 NEXT C
600 FOR C=1 TO B
610 IF A(C+10)=0 THEN 630
620 PRINT C
630 NEXT C
640 PRINT "FIN"
650 END
```

Le carré magique

Un carré magique se compose de nombres disposés d'une façon particulière dans les cases d'un carré. Que l'on additionne ces nombres en ligne, en colonne ou en diagonale, le résultat est toujours le même. Reportez-vous à l'exemple ci-contre : les colonnes, les lignes et les diagonales nous donnent toutes le même résultat : 15. Un ordinateur peut-il créer un carré magique ? A première vue, beaucoup répondront 'non'. Et pourtant, c'est très facile sur des carrés comportant un nombre impair de cases, comme par exemple 9, 25 ou 3969. L'auteur des règles permettant de composer des carrés magiques reste inconnu. On peut toutefois retrouver ces principes dans un des livres écrits par un maître en calcul : Adam RIESE (1492-1559). RIESE a précisé les règles permettant de construire un carré magique de rang impair et ces règles sont facilement programmables sur ordinateur.

4	9	2
3	5	7
8	1	6



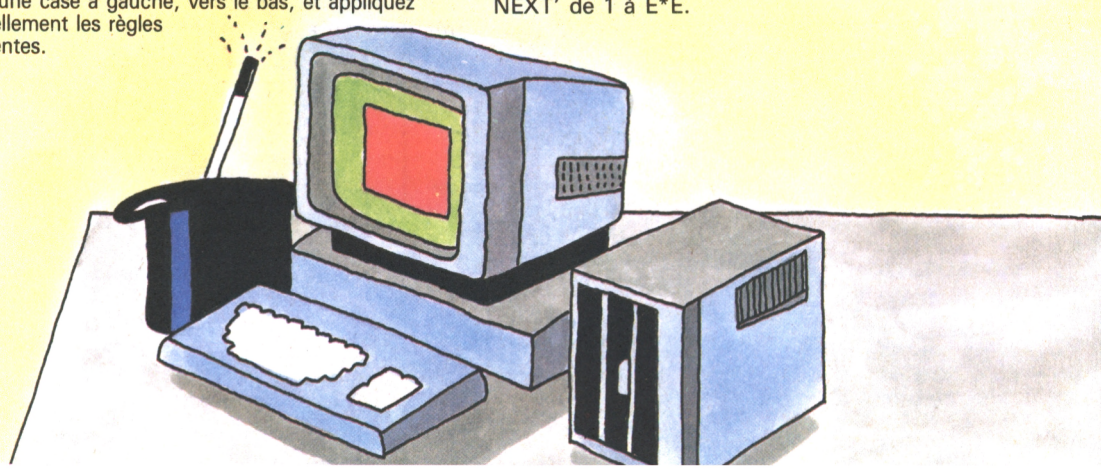
Règles de Riese pour la construction d'un carré magique

A. La construction commence à l'élément situé sous la case occupant le centre du carré (c'est toujours possible puisque nous nous intéressons aux carrés comportant un nombre impair de cases). Nous y plaçons le chiffre 1.
B. Le chiffre suivant se place à la case située en-dessous, à droite de la précédente, si cette case n'est pas déjà occupée. C'est la règle dite "sud-est".
C. Si cela vous conduit à sortir du carré vers le bas, vous êtes dans la bonne colonne mais pas dans la bonne ligne : occupez la case placée dans cette colonne, tout en haut du carré.
D. Si cela vous conduit à sortir du carré vers la droite, vous êtes sur la bonne ligne, mais pas dans la bonne colonne : occupez la case placée dans cette ligne, dans la première colonne du carré.
E. Si la case où vous arrivez est déjà occupée, décalez-vous d'une case à gauche, vers le bas, et appliquez éventuellement les règles précédentes.
F. Si, en suivant la règle 'B', vous êtes amené à sortir du carré suivant une diagonale, placez-vous dans la dernière colonne du carré, sur la deuxième ligne en partant du haut.

Ces règles restent toujours valables.

Le carré et l'ordinateur

Pour pouvoir construire et mémoriser un carré magique de cinq cases de côté, il est nécessaire d'avoir 25 cases en mémoire. Le plus simple est d'utiliser un tableau possédant 25 cases. Si nous appelons 'A' ce tableau, A(17) représente le dix-septième nombre. Si 'N' = 3, A(N) désigne la troisième case, donc le troisième nombre du tableau. Le programme vérifie si la taille du carré n'est pas trop grande et si le nombre total de cases est un carré parfait. Puis il initialise toutes les cases à zéro et applique les six règles. Cela donne une très grande boucle 'FOR ... NEXT' de 1 à E*E.

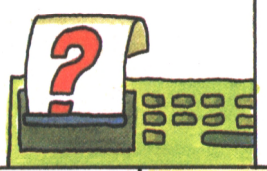


Renseignements complémentaires sur le programme

Au début du programme, à la ligne 250, l'ordinateur vérifie que la longueur du côté est inférieure à 10 et impaire (ligne 225). Entre les lignes 300 et 325, le tableau est rempli de zéros. L'adresse 'G' de chaque case sera déterminée par un compteur pour les lignes ('I') et un autre pour les colonnes ('C') (ligne 310). Ainsi, les lignes se rempliront les unes après les autres. A(1) est l'élément en haut à gauche dans le carré ; A(E*E), l'élément en bas à droite. Les lignes 330 à 530 remplissent le carré. D'après la règle 'A', on commence par l'élément sous le centre. Ce sont les lignes 330-331. Une boucle 'FOR ... NEXT' fonctionne avec un compteur 'F' (lignes 350-530).

Les cinq règles de 'B' à 'F' sont appliquées aux lignes suivantes :

Règle B : lignes 415 à 440. Règle C : ligne 450. Règle D : ligne 490. Règle E : lignes 510 à 515. Règle F : lignes 460 à 470. Le carré sera affiché par les lignes 600 à 680.



DIM et les matrices

Un carré de nombres s'appelle "matrice". C'est un tableau à deux dimensions. Les langages BASIC, dans leur grande majorité, sont capables de traiter les tableaux à deux dimensions. Pour cela, on doit réserver la place en mémoire par l'instruction 'DIM' ; par exemple, pour la matrice 'A', on écrira au début du programme :

```
DIM A(5,8)
```

ce qui détermine une matrice rectangulaire de 5*8 cases, soit 40 cases. Si l'on considère que le premier chiffre représente le nombre de lignes, le second chiffre le nombre de colonnes, et si l'on appelle respectivement ces nombres 'M' et 'N', on peut écrire :

```
M = 5 N = 8 DIM A(M,N)
```

Si l'on veut utiliser l'élément placé dans la troisième ligne, cinquième colonne, on écrira A(3,5).

Test des carrés d'Adam Riese

Là encore, ce programme fait plus appel à la logique qu'au calcul. Il faut donc le tester soigneusement. Pour cela, choisissez un carré de 5 de côté. Vous devez obtenir le carré suivant :

11	24	7	20	3
4	12	25	8	16
17	5	13	21	9
10	18	1	14	22
23	6	19	2	15

Si vous obtenez ce résultat, votre programme fonctionne correctement.

Le programme

```
100 DIM A(100)
200 REM CARRES MAGIQUES
210 PRINT "DONNEZ LE NOMBRE (INFE
    RIEUR A 10) DE"
211 PRINT "CASES PAR COTE :"
215 INPUT "";E
220 IF E<10 THEN 235
225 PRINT "NOMBRE TROP GRAND !"
230 GOTO 210
235 I=E-INT(E/2)*2
240 IF I<>0 THEN 300
245 PRINT "LE NOMBRE EST PAIR !"
250 END
300 FOR I=1 TO E
305 FOR C=1 TO E
310 G=((I-1)*E)+C+10
315 A(G)=0
320 NEXT C
325 NEXT I
330 I=INT(E/2)+2
331 C=INT(E/2)+1
350 FOR F=1 TO (E*E)
355 G=((I-1)*E)+C+10
360 IF I>E THEN 450
400 IF C>E THEN 490
410 IF A(G)<>0 THEN 510
415 A(G)=F:PRINTF,G
420 I=I+1
430 C=C+1
440 GOTO 530
450 IF C<=E THEN I=1:GOTO 355
460 C=E
470 I=2
480 GOTO 355
490 IF I<=E THEN C=1:GOTO 355
510 I=I+1
515 C=C-1
520 GOTO 355
530 NEXT F
600 FOR I=1 TO E
610 FOR C=1 TO E
620 G=((I-1)*E+C+10)
630 PRINT "LIGNE";I;" ";"COLONNE";
    C;" ";"VALEUR";A(G)
640 NEXT C
650 PRINT " "
660 NEXT I
670 PRINT "FIN."
680 END
```

Si vous avez une calculatrice de poche

Notre programme a été conçu pour pouvoir fonctionner sur une calculatrice de poche capable de stocker un peu plus de 1 400 pas de BASIC. Les pas inutilisés servent à la mémorisation des variables. Une variable utilise, sur les petites machines de poche, huit pas de programmes. On peut donc utiliser autant de variables qu'on le souhaite, dans la limite de ce que nous laisse comme place disponible le programme. Comme nous avons besoin de variables dans notre programme, nous utiliserons les noms 'A' à 'K'. Toutefois, le fonctionnement d'une calculatrice est tel qu'il y aurait confusion entre les variables 'C', 'E', etc., et les cases du tableau A(3), A(5), etc. Pour cela, les indices du tableau commencent à A(11). En conséquence, nous utilisons de temps à autre la valeur 10 (lignes marquées d'une étoile dans le programme) pour bien positionner les indices à partir de 11 et non pas de 1.

Notre minuscule calculatrice de poche possède encore environ 1 000 pas de mémoire libre, ce qui est suffisant pour un carré de 9 cases de côté, comportant 81 nombres.

Que peut-on faire d'autre ?

Le programme d'impression (lignes 600 à 680) présente l'avantage de pouvoir imprimer un carré quelle que soit la valeur de 'E'. Bien sûr, pour les gros carrés, on peut avoir à mettre en forme le carré. On pourrait également l'imprimer directement, ligne par ligne, au lieu de colonne par colonne. Cette modification est simple à réaliser.

On peut intervertir lignes et colonnes. On obtient ainsi un carré magique symétrique de l'original.

Un carré tient sur un seul écran jusqu'à environ 15 cases de côté. Si votre ordinateur a des possibilités graphiques, vous pouvez dessiner le carré avec toutes ses cases. Les lignes suivantes calculent la valeur magique du carré (c'est-à-dire sa somme caractéristique) :

```
655  H = (E*E*E+E)/2
666  PRINT "LE NOMBRE MAGIQUE EST";H
```

Jeu de la vie

Voici un problème de biologie : la vie et la mort d'une espèce, et son évolution. Prenons par exemple 'N' animaux vivant à une période donnée. Leur taux de natalité est de 25 %. En conséquence, G = 0.25*N animaux viendront au monde à chaque période. La mort est plus complexe à formuler. En effet, les animaux se gênent mutuellement car la nourriture qui est à leur disposition est limitée ; ils peuvent être touchés par des maladies contagieuses et s'entre-dévorent. Le pourcentage des animaux qui meurent se calcule en partant du nombre total des animaux vivants auquel on applique un facteur de mortalité 'B' :

S = 0.1 + B*N

Le total des morts à une période donnée est donc :

T = S*N = (0.1 + B*N)*N

A la fin de l'année, le nombre total des animaux s'élève à :

M = N + G – T

La définition des 'périodes' varie selon les animaux. Cela peut correspondre à une année ou à une heure (cas des bactéries).

Le modèle animal

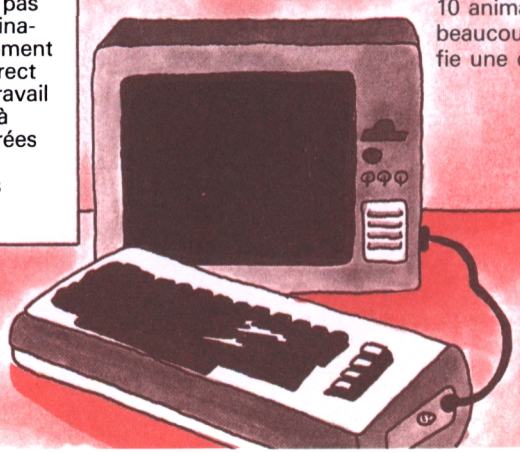
notre modèle tient compte du nombre total d'animaux au début de la première période, du facteur de handicap 'B' et du nombre total de périodes sur lesquelles sont effectués les calculs. A partir de cela, le programme calcule le nombre 'M' des animaux à la fin de la première période et repart de cette valeur 'M' pour la prochaine période, une fois que le numéro de la période et le nombre d'animaux sont imprimés. Dans le cas où 'M' ne change pas d'une période à l'autre, on dit que la population de cette espèce est constante. Le programme s'arrête alors de lui-même. Si le nombre d'animaux descend en dessous de zéro, le programme annonce "espèce éteinte". Le programme est relativement simple, mais il n'est pas facile de déterminer B. Lorsque l'ordinateur n'existait pas, il était particulièrement compliqué de trouver un résultat correct avec de telles formules, car tout le travail de calcul se faisait à la main. Grâce à l'ordinateur, les approximations chiffrées sont devenues, dans le domaine des sciences naturelles, une pratique très banale.

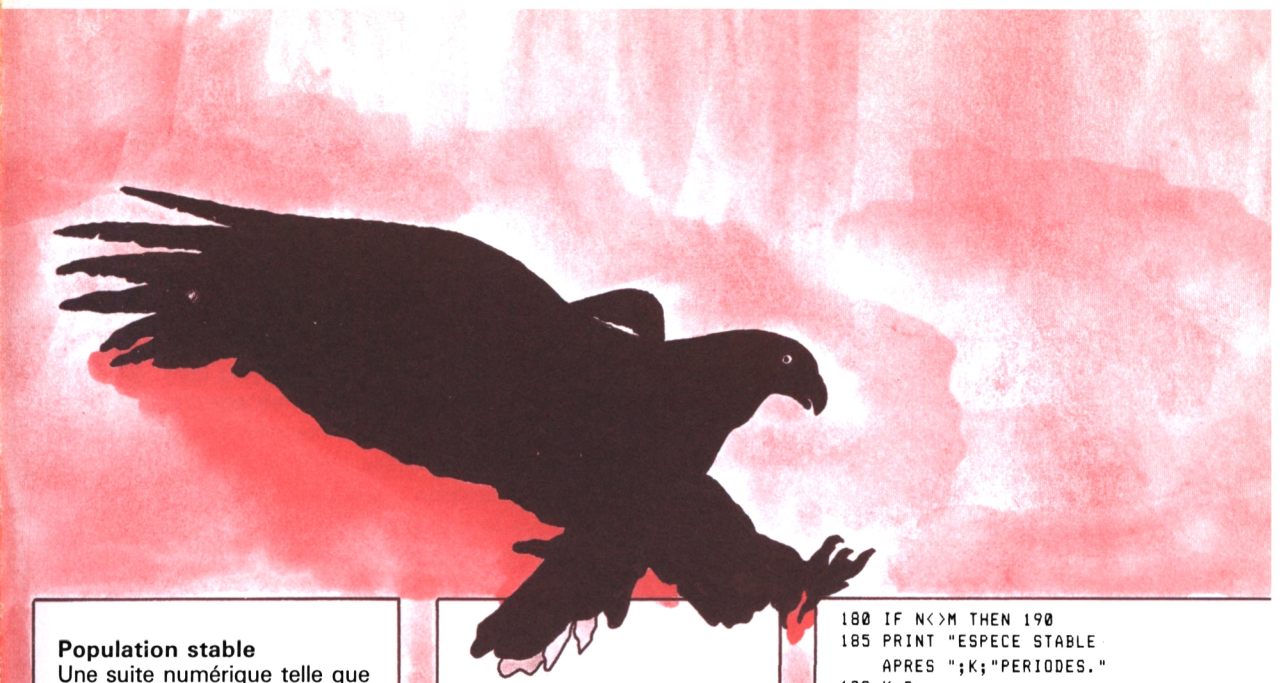
Le programme du jeu de la vie

Le programme est relativement simple. Il lui est toutefois nécessaire de connaître les valeurs que vous choisissez pour N et B, sinon il ne pourrait mener à bien ses calculs. B et N ont une grande importance. Ils sont essentiels pour le fonctionnement du programme, car ils déterminent l'évolution de l'espèce. Le taux de mortalité se calcule grâce à la formule suivante :

S = 0.1 + B*N

Si l'on veut que l'espèce se maintienne, il faut impérativement que S soit inférieur à 1, sinon il mourrait plus d'animaux qu'il y en a au total. Or, pour cela, il faut que B*N soit inférieur à 0.9. D'autre part, il faut aussi qu'il ne meure pas plus d'animaux qu'il n'en naît. Si, maintenant, on écrit B*N = 0.15, la natalité équilibre la mortalité. On ne peut obtenir cet état de fait que dans la mesure où B et N restent constants. C'est là que réside le problème. Pour 1 000 animaux, B doit se situer aux alentours de 0,0001. Pour 100 animaux, B doit avoisiner 0,001, et 0,01 pour 10 animaux. Si B est petit pour beaucoup d'animaux, cela signifie une croissance rapide.





Population stable
Une suite numérique telle que la calcule le programme peut évoluer de manières très différentes. Dans notre cas, les nombres peuvent augmenter. L'espèce se développe donc, mais le taux de mortalité s'élève également. Les valeurs des nombres peuvent baisser, ce qui entraîne simultanément une baisse de la mortalité. Par la suite, les chiffres pourraient éventuellement remonter. Enfin, la suite numérique peut se stabiliser ou osciller très légèrement autour d'une valeur. Dans ce cas, l'espèce se stabilise et le programme s'interrompt lorsque le même résultat est obtenu sur deux périodes consécutives. Le programme s'arrête également lorsqu'il n'y a plus d'animaux.

Le programme

```
10 REM EVOLUTION D'UNE ESPECE
20 INPUT "NOMBRRE TOTAL D'ANI
   MAUX ";N
25 PRINT "LE NOMBRE TOTAL D'A
   NIMAUX EST : ";N
30 INPUT "TAUX DE HANDICAP ";B
35 PRINT "LE TAUX DE HANDICAP
   EST :";B
40 INPUT "NOMBRE DE PERIODES";P
45 PRINT "LE NOMBRE DE PERIO
   DES EST ";P
95 M=N
100 FOR K=1 TO P
110 N=M
120 G=.25*N
130 S=.1+B*N
140 T=S*N
150 M=INT(N+G-T)
160 IF M>0 THEN 170
165 PRINT "ESPECE ETEINTE
    APRES ";K;" PERIODES.":K=P
168 GOTO 190
170 PRINT K;" ";M
180 IF N<>M THEN 190
185 PRINT "ESPECE STABLE
    APRES ";K;"PERIODES."
188 K=P
190 NEXT K
200 PRINT "FIN."
210 END
```

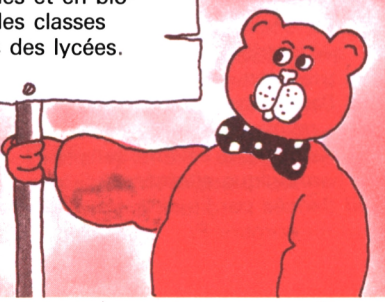
Test
Le mieux est de fournir au programme un exemple à calculer après que l'on a tapé et vérifié chaque ligne. Voici des données qui conduisent rapidement à une situation stable :
Nombre total d'animaux : 130
Taux de mortalité : 0.00105
Total des périodes : 25

1	131
2	132
3	133
4	134
5	135
6	136
7	136

Le nombre d'animaux se stabilise en sept périodes.

Autres utilisations du modèle
Il est bien sûr possible de modifier le taux de natalité et de mortalité de notre programme. Au départ, il est de 0,25. Il est simple de modifier le taux de natalité sans changer le nombre d'animaux et d'essayer de comprendre comment le modèle réagit devant d'autres chiffres. Pour l'amateur éclairé, il est particulièrement intéressant de représenter graphiquement les nombres d'animaux obtenus en fonction de ces modifications. Pour cela, il est cependant nécessaire de disposer d'un ordinateur plus important qu'une simple calculatrice de poche. Voici un exemple : dix animaux avec un taux de mortalité de 0,00105 se multiplient en 45 étapes environ pour atteindre le nombre stable de 136. On peut également dire qu'il existe dix millions de bactéries qui grandissent en des millions d'étapes et qui, en 45 heures, atteignent le nombre total de 136 millions.

Remarque
On peut retrouver l'exposé de ce problème, ainsi que celui des renards et des lapins, dans des livres destinés à l'enseignement en sciences naturelles et en biologie pour les classes supérieures des lycées.



Cinq fois l'infini : suites et séries

Le mathématicien manipule souvent des suites infinies de nombres ainsi que leurs sommes. Voici un exemple : 1, 1/2, 1/4, 1/8, 1/16... constituent une suite dont chaque terme est déduit du précédent par une division de moitié. Qui aurait suffisamment de temps pour calculer la somme de ces termes :
S = 1 + 1/2 + 1/4 + 1/8 + 1/16... ?
La tâche n'est pas compliquée mais répétitive. On peut toutefois remarquer que la valeur du nième élément de cette somme tend vers 0 lorsque n devient grand. En conséquence, après avoir sommé un certain nombre d'éléments, S ne variera pratiquement plus quel que soit le nombre de termes ajoutés ensuite. On dit que cette somme converge. Il existe des sommes telles que le nième élément tend vers 0 mais influe encore sur le résultat : la somme elle-même évoluera sans cesse ; on dit que la somme diverge ; aussi loin que l'on pousse le calcul, le résultat ne se stabilisera pas.

Différentes séries

Il existe un très grand nombre de suites infinies, réparties grossièrement en trois catégories : les suites divergentes (dont on ne pourra jamais calculer la somme), les suites à convergence rapide (il suffit de calculer peu d'éléments pour obtenir la somme avec une bonne précision) et les suites à convergence lente (il faut calculer des centaines, voire des milliers de termes pour obtenir un résultat fiable).
La théorie des séries est ardue mais passionnante. Nous nous intéresserons ici à cinq séries particulières que nous demanderons à l'ordinateur de calculer. Ce travail reste également à la portée d'une calculatrice de poche fonctionnant en BASIC.

Dénomination des cinq programmes

GE01 (lancée par l'instruction 'RUN 5') : série géométrique comprenant 'N' termes.
GE02 (lancée par l'instruction 'RUN 105') : série géométrique calculée avec une limite sur la précision.
GE03 (lancée par l'instruction 'RUN 205') : série harmonique divergente.
GE04 (lancée par l'instruction 'RUN 305') : série harmonique alternée.
GE05 (lancée par l'instruction 'RUN 405') : calcul du nombre Pi.

La suite géométrique

La somme S = 1 + 1/2 + 1/4 + 1/8 + 1/16 + ... converge assez rapidement. Elle s'appelle géométrique car on peut passer d'un terme de la suite au terme suivant par un coefficient multiplicateur.
Le programme GE01 calcule 'S1' pour 'N' termes, 'N' étant donné par l'utilisateur. Une boucle 'FOR ... NEXT' est utilisée pour cela, le compteur de boucle variant de 1 à 'N'. Cette suite correspond à un problème bien connu : Un garde forestier rentre chez lui : son chien court deux fois plus vite. Arrivé à la maison, le chien repart à la rencontre de son maître, puis, arrivé vers son maître, il repart vers la maison, et ainsi de suite. Finalement, lorsque le garde forestier arrive chez lui, quelle distance le chien a-t-il parcourue ?
Réponse : La distance parcourue par le garde multipliée par deux. La résolution de ce problème passe par l'utilisation de la suite 'S1' dont la somme vaut 2.

GE02 traite la même série ; cette fois, l'utilisateur ne donne pas le nombre 'N' d'éléments composant la somme, mais la précision 'E'. Dès que la valeur d'un élément deviendra inférieure à 'E', le programme s'arrêtera.

La série harmonique

Le programme GE03 traite la série harmonique :

S2 = 1 + 1/2 + 1/3 + 1/4 + 1/5 + ...

Le calcul s'effectue sur 'N' éléments. Cette série est divergente : à chaque nouvel élément, le résultat évolue. Le programme affiche le résultat après chaque calcul. Si vous ne le souhaitez pas, supprimez la ligne 235 et réécrivez-la en ligne 242. Seul le résultat final apparaîtra, lorsque 'N' calculs auront été effectués.

Série convergente vers Pi

Lorsque l'on connaît la limite vers laquelle tend la somme des éléments d'une série, l'utilisation même des éléments et le calcul de la somme permettent de déterminer cette limite avec plus de précision. Prenons un exemple : on démontre que l'expression 4*(1−1/3+1/5−1/7+1/9−1/11 ...) converge vers Pi. Pi est donc la limite de cette somme et le calcul de cette somme avec un grand nombre d'éléments nous permet de calculer Pi, avec autant de décimales qu'on le veut. Malheureusement, cette somme converge lentement. Il existe d'autres algorithmes nettement plus complexes mais convergeant plus rapidement.

Les programmes

```
5 REM GE01
15 INPUT "NOMBRE D'ELEMENTS ";N
20 S=1
25 G=1
30 FOR K=2 TO N
35 PRINT K;" ";S
40 G=G*.5
45 S=S+G
50 NEXT K
55 PRINT "FIN."
60 END
```

```
105 REM GE02
115 INPUT "LIMITE DE PRECISION ";E
120 PRINT "LA LIMITE EST ";E
125 S=1
128 N=1
130 G=1
135 G=G*.5
140 S=S+G
145 IF G<=E THEN GOTO 165
150 N=N+1
155 PRINT N;" ";S
160 GOTO 135
165 PRINT "FIN."
170 END

205 REM GE03
215 INPUT "NOMBRE D'ELEMENTS ";N
220 S=0
225 FOR K=1 TO N
230 S=S+1/K
235 PRINT K;" ";S
240 NEXT K
245 PRINT "FIN."
250 END

305 REM GE04
315 INPUT "NOMBRE D'ELEMENTS ";N
320 S=0
325 M=-1
330 F=-1
335 FOR K=1 TO N
340 M=M*F
345 S=S+(M/K)
350 PRINT K;" ";S
355 NEXT K
360 PRINT "FIN."
365 END

405 REM GE0 5
415 INPUT "NOMBRE D'ELEMENTS ";N
420 I=0
425 S=0
430 M=-1
435 F=-1
440 FOR K=1 TO 2*N STEP 2
445 I=I+1
450 IF I<=1000 THEN GOTO 465
455 PRINT ((K-1)/2);" ";P
460 I=1
465 F=F*M
470 G=F*(1/K)
475 S=S+G
480 P=4*S
485 NEXT K
490 PRINT "PI = ";P
495 END
```

Test

Comme les calculs sont ici nombreux, il est important de vérifier le bon fonctionnement du programme.

Voici les dix premières valeurs que vous devez obtenir pour le programme GE01 (en tapant 'RUN 5') :

```
NOMBRE D'ELEMENTS ? 10
2          1
3          1.5
4          1.75
5          1.875
6          1.9375
7          1.96875
8          1.98438
9          1.99219
10         1.99609
```

Si l'on choisit la limite 'E = 0.00001', on obtient pour GE02 (en tapant 'RUN 105') :

```
LIMITE DE PRECISION ? .00001
LA LIMITE EST 0.00001
2          1.5
3          1.75
4          1.875
5          1.9375
6          1.96875
7          1.98438
8          1.99219
9          1.99609
10         1.99805
11         1.99902
12         1.99951
13         1.99976
14         1.99988
15         1.99994
16         1.99997
17         1.99998
```

Voici les dix premières valeurs obtenues par le calcul de la suite harmonique divergente (en tapant 'RUN 205') :

```
NOMBRE D'ELEMENTS ? 10
1          1
2          1.5
3          1.83333
4          2.08333
5          2.28333
6          2.45
7          2.59286
8          2.71786
9          2.82897
10         2.92897
```

Les valeurs suivantes sont fournies par la série harmonique convergente (en tapant 'RUN 305') :

```
NOMBRE D'ELEMENTS ? 10
1          1
2          .5
3          .833333
4          .583333
5          .783333
6          .616667
7          .759524
8          .634524
9          .745635
10         .645635
```

Enfin, le programme débutant en 405 converge très faiblement ; il donne les résultats suivants :

```
Pour 'N' calculs   valeur de Pi
N = 1000           3,14059
N = 2000           3,14109 (Pi = 3,14159...)
```

Pour tous ces résultats, il est possible, notamment pour Pi, que les derniers chiffres affichés par votre ordinateur soient différents de ceux-ci en raison de la précision différente d'une machine à l'autre.

Série harmonique alternée

Le programme GE04 commence à la ligne 305. Il calcule la somme 'S3' :

S3 = 1 − 1/2 + 1/3 − 1/4 + 1/5 − 1/6 ...

Le mot "alternée" signifie que le signe change à chaque élément. Les éléments diminuent en absolu, mais si lentement que la série diverge. Toutefois, cette série alternée diverge plus lentement que la série harmonique vue précédemment. Vous pourrez le constater en laissant l'ordinateur calculer autant de fois que vous le désirez.

Dans ce programme, le problème consiste à changer le signe à chaque élément. Pour cela, 'M' et 'F' sont initialisés à '−1'. A chaque étape, 'M' est multiplié par 'F', donc 'M' change de signe à chaque fois.

L'ordinateur écrivain : est-ce possible ?

Il est difficile de déterminer si un ordinateur est capable ou non d'écrire des phrases tant qu'on ne sait pas comment fonctionne l'écriture automatique. Pourtant, il est facile de laisser l'ordinateur babiller tout seul, ce qui est très amusant car on ne peut pratiquement pas deviner ce qu'il va nous dire.
Déjà, dans les années cinquante, on savait faire raconter des histoires à un ordinateur. Le principe est simple : on fournit à la machine un exemple de phrase grammaticalement correcte et on lui demande de choisir au hasard un sujet parmi un ensemble de sujets pré-enregistrés, un verbe parmi un ensemble de verbes, etc.

En faisant fonctionner le programme, vous constaterez que les histoires qu'il fournit ont par instant une connotation très moderne. En voici quelques exemples :
TOUT HOMME EST FIDELE
TOUTE FEMME EST FERTILE
L'EAU EST INCOLORE
LE TERRE EST AVARE
Mis à part le mot ''est'', tous les mots sont tirés au hasard. Toutefois, l'accord du genre est respecté.

Construction de la phrase
Etudions la construction d'une phrase en prenant l'exemple :
TOUT HOMME EST FIDÈLE.
Le programme accepte huit sujets, comme 'homme', 'femme', 'jardin' et 'eau'..., huit adjectifs : 'fidèle', 'multicolore', 'fertile'... Le programme relie ces mots par le verbe 'est'. Enfin, avant chaque sujet, un autre adjectif ou un article est ajouté.

A quoi ressemble le programme ?
Pour initialiser le générateur de nombres tirés au hasard (voir pages 22, 23), le programme a besoin d'un nombre 'H' compris entre '0' et '0.3' donné par l'utilisateur. Le programme doit également savoir le nombre de lignes que doit comporter le texte. Ceci constitue la première partie du programme.
Pour la mise en mémoire de notre lexique, nous utiliserons un tableau qui portera le nom 'A$' car il doit stocker des données alphabétiques. Le premier mot sera placé dans la case A$(20). Le tableau se termine à A$(59) et contient tout le vocabulaire utilisé. Entre les lignes 100 et 145, le tableau est initialisé. Puis vient une large boucle, s'exécutant 'P' fois et débutant à la ligne 400. Cette boucle fabriquera quatre nombres aléatoires qui seront ensuite transformés en indices compris entre 20 et 59. A chacun de ces nombres aléatoires correspondra donc finalement un élément du tableau A$(N) et les quatre mots ainsi trouvés constitueront la phrase. L'article adéquat se trouve à chaque fois placé en mémoire juste avant le nom correspondant. En testant l'article lui-même pour retrouver son genre, on peut déterminer l'adresse correcte de certains mots comme 'tout', 'aucun', 'mon' ou 'notre'. La ligne ainsi constituée est imprimée.
Un générateur de nombres aléatoires est inclus dans le programme à partir de la ligne 900. Il est inutile si votre ordinateur reconnaît la fonction RND. Dans ce cas, ajoutez la ligne :

```
80 J = RND(0)
```

Un nombre aléatoire supérieur à 0 et inférieur à 1 est immédiatement placé dans la variable J. Le nombre de chiffres après la virgule dépend de la précision du langage BASIC utilisé. Ajoutez au début du programme :

```
15 RANDOMIZE
```

pour initialiser le générateur différemment à chaque exécution. Si votre ordinateur ne connaît pas ces instructions, ne modifiez pas le programme.

Le programme

```
10 REM HISTOIRE
20 INPUT "TAPEZ UN NOMBRE ENTRE 0
   ET 0.3 ";H
40 Q=H+0.314159625
50 INPUT "COMBIEN DE LIGNES ";P
70 DIM A$(60)

100 A$(20)="L'"
101 A$(21)="HOMME"
102 A$(22)="LA"
103 A$(23)="FEMME"
104 A$(24)="L'"
105 A$(25)="ENFANT"
106 A$(26)="LA"
107 A$(27)="MAISON"
108 A$(28)="LE"
109 A$(29)="JARDIN"
110 A$(30)="LA"
111 A$(31)="FLEUR"
112 A$(32)="L'"
113 A$(33)="EAU"
114 A$(34)="LA"
115 A$(35)="TERRE"
120 A$(36)="INSIPIDE"
121 A$(37)="AVARE"
122 A$(38)="FERTILE"
123 A$(39)="PROSPERE"
124 A$(40)="PROCHE"
125 A$(41)="INCOLORE"
126 A$(42)="MULTICOLORE"
127 A$(43)="FIDELE"
130 A$(44)="CHAQUE"
131 A$(45)="CHAQUE"
132 A$(46)="CHAQUE"
133 A$(47)="TOUT"
134 A$(48)="TOUTE"
135 A$(49)="TOUT"
136 A$(50)="MON"
137 A$(51)="MA"
138 A$(52)="MON"
139 A$(53)="NOTRE"
140 A$(54)="NOTRE"
141 A$(55)="NOTRE"
142 A$(56)="ET"
143 A$(57)="OU"
144 A$(58)="MAIS"
145 A$(59)="."

400 FOR K=1 TO P
410 GOSUB 900
415 IF J>7 THEN GOTO 410
420 B=J*2+20
430 GOSUB 900
435 IF J>7 THEN GOTO 430
440 C=J+36
450 GOSUB 900
460 IF J>3 THEN GOTO 450
490 IF J=0 THEN D=B:GOTO 510
500 D=J*3+41
510 GOSUB 900
520 IF J>3 THEN GOTO 510
550 E=J+56
555 IF D=B THEN GOTO 580
560 IF A$(B)="LA" THEN D=D+1:GOTO
    580
570 IF A$(B)="LE" THEN D=D+2:GOTO
    580
580 B=B+1
585 PRINT A$(D);" ";A$(B);" EST "
    ;A$(C);" ";A$(E)
590 NEXT K
610 END

900 IF Q>=.4 THEN GOTO 902
901 Q=Q+.35
902 Q=Q*Q
903 R=INT(Q*100000)
904 S=INT(Q*10000)*10
905 J=R-S
906 RETURN
```

La grammaire de l'ordinateur

La grammaire que possède votre ordinateur est d'une grande simplicité. Si vous souhaitez produire des phrases plus sophistiquées, vous rencontrerez de nombreuses difficultés, car la grammaire et la syntaxe du français ne sont pas simples à respecter. Vous en déduirez qu'il faut encore beaucoup de réflexions et d'études avant que l'ordinateur sache vraiment écrire. Par ailleurs, cela demande des machines plus puissantes qu'une simple calculatrice.

Commentaires

La structure du programme est assez simple. Seul le calcul des adresses dans le tableau 'A$' est un peu plus compliqué.
Le programme se divise en quatre grandes parties :

1. Le générateur de nombres aléatoires est mis en place de la ligne 10 à la ligne 60. On devra fournir le nombre total 'P' de lignes à écrire.
2. Entre les lignes 100 et 145, le vocabulaire est chargé dans les tableaux 'A$'.
3. Entre les lignes 400 et 610, les nombres aléatoires sont transformés en indices servant à trouver les mots dans 'A$'. Puis ces mots seront imprimés dans l'ordre voulu (ligne 585).
4. Entre les lignes 900 et 906, nous trouvons un sous-programme comportant le générateur de nombres aléatoires appelé par l'instruction 'GOSUB'.

Test

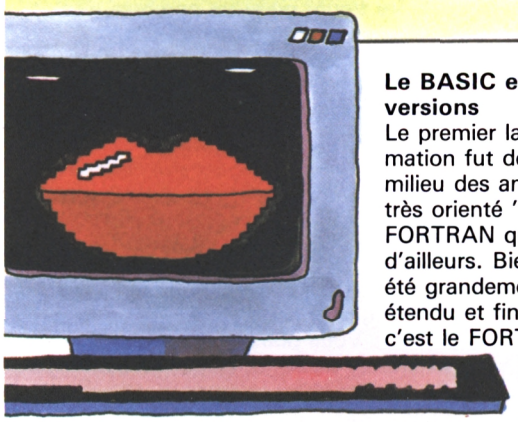
Le test n'est pas simple à réaliser. Les nombres aléatoires utilisés doivent être compris dans des fourchettes étroites : entre 0 et 7 par exemple pour les sujets, parfois même entre 0 et 3.
Le programme d'écriture cherche un chiffre aléatoire par l'intermédiaire de l'instruction 'GOSUB'. L'instruction 'IF' vérifie la validité du résultat. Si le chiffre ne correspond pas à ce qu'on attend, un autre chiffre est demandé. Pour simplifier les tests, le mieux est de faire afficher par une instruction 'PRINT' les trente premiers nombres aléatoires produits par le générateur, et de lancer le programme avec le générateur initialisé toujours de la même manière (variable 'H'). Connaissant les trente premières valeurs que le programme utilisera, vous pouvez ainsi vérifier son bon fonctionnement en retrouvant les mots qu'il doit logiquement employer.

Comment aller plus loin ?

Nous avons déjà dit quelques mots sur la possibilité d'améliorer le programme en lui faisant reconnaître d'autres règles de grammaire. Si votre ordinateur est suffisamment puissant, vous pourrez également programmer d'autres types de phrases. Vous pouvez, à l'aide du générateur de nombres aléatoires, allonger les phrases en ajoutant également d'autres mots. Vous pouvez aussi changer les mots pour en utiliser de plus poétiques. Vous serez étonné de constater combien le changement de mots influe sur le résultat : la différence est grande entre un roman policier et un conte de fée. Enfin, vous pouvez augmenter le vocabulaire mémorisé par votre machine.

Comment travailler sur un programme

Si vous avez fait fonctionner l'un ou l'autre des douze programmes contenus dans ce livre, et si vous l'avez amélioré, vous avez alors découvert un certain nombre de notions. Par exemple, vous avez certainement constaté que rien n'est simple sur un programme que l'on n'a pas écrit soi-même : on se sent un peu perdu, on ne comprend pas pourquoi cela a été conçu de cette manière et on ne sait pas à qui se plaindre lorsqu'on rencontre une difficulté.
Lorsqu'on écrit des programmes et qu'on tente de les faire fonctionner, on arrive généralement à un point où l'on constate qu'il vaudrait mieux tout mettre à la poubelle pour tout recommencer. Mais qui vous assure que le second essai fonctionnera mieux que le premier ? Il ne vous reste plus qu'à persévérer, coûte que coûte. Heureusement, on s'en sort toujours. C'est pour cette raison que presque chaque programme comporte des ajouts révélant à un professionnel les parties que vous avez beaucoup travaillées. Par exemple, c'est une suite étrange de numéros de lignes, à moins qu'il ne s'agisse de noms de variables exotiques. Dans notre cas, nous avons sciemment renoncé à procéder à une dernière rectification de tous nos programmes.



Le BASIC et ses diverses versions
Le premier langage de programmation fut découvert vers le milieu des années 50. Il était très orienté "Calculs" : c'est le FORTRAN qui existe toujours d'ailleurs. Bien sûr, ce langage a été grandement amélioré, étendu et finalement normalisé : c'est le FORTRAN IV des années 60. En 1963, un groupe d'universitaires américains développa un autre langage à partir du FORTRAN. On le nomma BASIC (Beginner's All purpose Symbolic Instruction Code). A l'origine, le BASIC comportait deux douzaines de mots clés découlant presque tous du FORTRAN. Il pouvait fonctionner sur des ordinateurs relativement petits. Depuis, sont nées de nombreuses versions plus sophistiquées.
A celui qui s'intéresse à l'informatique, nous conseillons d'apprendre un second langage en plus du BASIC, ceci afin de ne pas rester limité aux faiblesses et aux inconvénients du langage BASIC.

Différents langages BASIC
Nos programmes sont rédigés dans un BASIC élémentaire sans grandes astuces. Mais un langage BASIC réclamera un point-virgule là où un autre utilisera une virgule. Lorsque vous lancez l'exécution du programme, si l'ordinateur ne comprend pas une ligne, il vous signale l'erreur à l'écran et affiche le numéro de ligne qui lui déplaît. Certains n'inscrivent que le numéro, d'autres affichent la ligne complète. Dans ce cas, il faut relire la ligne désignée caractère par caractère, en cherchant l'erreur. S'il n'y en a pas, vous devez regarder dans votre manuel de BASIC, pour vérifier si l'instruction existe et si la syntaxe est bonne. Si vous découvrez une variante de langage, il suffit de corriger toutes les autres erreurs du même genre dans le programme.

IF
Dans certains cas, IF peut être utilisé avec une autre syntaxe que la nôtre. Voici les formes que nous utilisons :
1. Comparaison logique ou numérique :

```
IF B + 3 = C THEN GOTO 50
ou IF B + 3 = C GOTO 50
ou IF B + 3 = C THEN 50
ou IF A = B THEN B$ = B$+C$
```

Tests
Si votre ordinateur est équipé d'aides à la mise au point (par exemple s'il comprend les instructions DEBUG ou TRACE), n'hésitez pas à vous en servir. Grâce à ces outils, vous pourrez faire exécuter le programme pas à pas et examiner le contenu des variables autant de fois que vous le désirerez.
Si l'instruction INPUT est utilisée dans le programme, tapez d'abord les valeurs que nous vous indiquons dans le paragraphe "TEST" correspondant. Vous devez obtenir les mêmes résultats.
Le mode TRACE (ou TRON/TROFF sur Thomson) affiche les numéros des lignes exécutées. Vous détectez ainsi le moment où une boucle ne se finit plus, où un test ne s'effectue pas correctement, etc. Vous pouvez également placer dans le programme des instructions PRINT supplémentaires pour faire afficher le contenu des variables les plus importantes. Lorsque le programme fonctionne, vous éliminez ces lignes. N'oubliez pas d'être très logique dans ces phases de test.

2. Comparaison entre des chaînes alphanumériques :
   IF variable alphanumérique = variable alphanumérique... :
   IF A$ = "ABC"
   ou IF A$ = B$

Si cette version de IF n'est pas comprise par votre ordinateur, il faut changer l'écriture de la ligne et son contenu.

LET
Certains ordinateurs exigent le mot LET devant le symbole d'attribution =. Dans notre cas, ce n'est pas nécessaire.

DIM
Le BASIC de certains micro-ordinateurs ne connaît pas l'instruction DIM. On peut donc délibérément utiliser un tableau A(n) sans cette instruction. La taille maximale du tableau dépend de la place disponible en mémoire. La plupart des langages BASIC exigent DIM lorsqu'une variable indicée doit contenir plus de dix valeurs.

Le programme est trop long
Il est pénible de s'apercevoir qu'un programme ne tient pas en mémoire. Les programmes de ce livre ont une longueur telle qu'ils doivent pouvoir fonctionner sur tous les micro-ordinateurs. Les paragraphes ci-dessous concernent les micros de poche comme par exemple le SHARP PC 1211.
Dans ce type d'ordinateur de poche, 26 cases mémoire contiennent les valeurs associées aux noms des variables A à Z.
Dans le programme, ces cases s'appellent indifféremment A, A$, A(1) ou A$(1). La case désignée est la même mais son contenu n'est pas interprêté de la même manière selon le type de variable utilisée (variable numérique, alphanumérique, tableau numérique, tableau alphanumérique).
Le logiciel peut comporter au maximum 1 424 pas de programme. Un pas correspond à une instruction simple. Pour un ordre compliqué, le processeur utilise plusieurs pas.
Mais la mémoire peut également servir à enregistrer des données. Une variable occupe alors l'équivalent de 8 pas de programme et ces variables viennent en complément des 26 premières déjà citées. La 27e variable utilisée est donc la première à empiéter sur la zone des 1424 pas de programme. On peut augmenter le nombre de variables jusqu'à 204 en théorie mais il n'y a plus alors de place pour le programme lui-même. Cette "cloison mobile" entre le programme et les variables au-delà de la 26e est automatiquement gérée par le microprocesseur de l'ordinateur.
Si le programme est trop grand, il faut soit renoncer à le résoudre, soit sacrifier des parties du programme. On élimine par exemple toutes les remarques (REM). On remplace ensuite, si c'est possible, tous les messages et chaînes alphanumériques par de simples initiales ou codes. Si cela ne suffit pas, il faut reprendre le programme lui-même et chercher à le réduire, par exemple en utilisant de meilleurs algorithmes plus concis. Si cela ne suffit toujours pas, il faut envisager le fractionnement du programme en plusieurs parties s'enchaînant. Afin de connaître la place occupée en mémoire, vous pouvez utiliser sur le SHARP PC 1211 la commande MEM. L'ordinateur répond alors :

384 STEPS 48 MEMORIES

Les "Steps" correspondent aux pas utilisés par le programme ; les "Memories" correspondent aux variables utilisées.
Le développement de la micro-informatique est tel que l'on peut enregistrer de plus en plus de données dans un volume de plus en plus restreint. Chaque nouvelle génération de machine repousse nettement les limites de la génération précédente. On peut aussi étendre la capacité de certaines machines en ajoutant soi-même des modules d'extension mémoire.

Le programme est trop lent
Raccourcir un programme ne constitue pas une tâche simple. L'accélérer est encore plus difficile.
Un programme peut "boucler", c'est-à-dire qu'une erreur a pu se glisser dans une ligne et une boucle FOR ... NEXT ou IF ... GOTO s'exécute indéfiniment. En voici un exemple :

```
10  FOR K = 1 TO 10
20  K = K - 1
30  NEXT K
```

Lorsqu'un programme fonctionne trop lentement à votre goût, recherchez d'abord la partie responsable de cette lenteur. Placez divers ordres PRINT dans le programme et chronométrez : vous situerez rapidement les phases laborieuses du logiciel.
Placez dans une boucle le minimum d'instructions nécessaires afin de ne pas l'allonger inutilement. Vérifiez le nombre de fois que la boucle s'exécute. Faites afficher les valeurs intermédiaires, les compteurs de boucles et les variables importantes par des instructions PRINT que vous supprimerez lorsque la mise au point sera terminée. Vous vérifierez ainsi que les boucles s'exécutent correctement.

Une bonne idée

Vous connaissez peut-être dans votre entourage, au lycée, dans votre voisinage ou même parmi vos proches, des personnes qui aiment travailler sur un ordinateur, des personnes qui acceptent plus volontiers un programme qu'une boîte de chocolats ! Nous serions charitables en vous conseillant de leur offrir notre livre, mais cela nous semble une bien meilleure idée de leur offrir une disquette ou une cassette sur laquelle vous aurez enregistré nos programmes après les avoir fait fonctionner sur votre propre ordinateur. Il est bien entendu qu'à notre avis le livre que vous lisez actuellement, ainsi que d'autres de la même série, font partie intégrante de ce cadeau.

Bibliothèque de programmes

Nous avions un peu l'intention, par l'intermédiaire de ce livre, de vous convaincre que l'on ne doit pas tout programmer soi-même. Certains programmes existent déjà dans le commerce, alors pourquoi ré-inventer la roue ? Vous pouvez acheter des programmes. Certains sont chers et très professionnels. Ce livre vous fournit douze programmes clés et peut vous en suggérer d'autres. La personne qui possède de nombreux programmes peut se constituer une bibliothèque de programmes, c'est-à-dire des disquettes et des cassettes sur lesquelles se trouvent des programmes dûment testés et vérifiés, prêts à l'emploi.

L'organisation de la bibliothèque dépend de votre ordinateur. Nous vous conseillons toutefois de ranger dans votre bibliothèque tous les programmes qui vous plaisent dans ce livre. Une bibliothèque de programmes testés évite la recherche d'erreurs et vous fournit un ensemble de programmes fiables, utilisables à tout instant par votre ordinateur.

La documentation

Une bibliothèque de programmes ne devrait jamais comporter de programmes qui n'aient été correctement testés. A chaque programme doivent correspondre des notes précises, expliquant le contenu du programme et la manière de le tester. On appelle ces notes la "documentation". Des programmes sans documentation restent sans valeur. Voici les éléments de base d'une bonne documentation :

* Un listing complet de la dernière version du programme, tel que vous l'obtenez par l'instruction LIST.
* Une description générale du programme dans ses grandes lignes.
* Des conseils et des données permettant de tester le programme.
* Des indications sur la marche à suivre pour charger le programme.
* Des indications et des commentaires sur les caractères plus spécifiques du programme.

Nous avons tenté de concevoir ce manuel de manière à ce qu'il représente une documentation utile pour les programmes qu'il fournit.

Index

A
Affichage, affiché 9
Agenda téléphonique 16
Aléatoire 14
Allumette 20
Alphanumérique 8
Alternée (suite) 31
Animaux 28

B
BASIC 6, 8
Bibliothèque 36
Bissextile 13
Boucle 9

C
Calcul d'adresse 12
Calendrier 12
Carré magique 26
Chaîne de caractères 8
Chargement en mémoire 35
Circulation routière 22
CLEAR 6
Commentaires 9
Compteur de boucles 9, 16
Constante 9
Construction de phrase 32
Convergente (série) 30
Crible 24

D
Date 12
DEBUG 7
Déterminer 7
Deviner 18
Différences entre Basic 34
DIM 25
Dimension 25
Divergente (série) 30
Documentation 36
Donnée 8

E
Écrivain 32
END 9
Ératosthène 25
Erreur 7
Expression 9

F
Facteur de gourmandise 10
Faute de frappe 9
Fin de programme 16
FOR 9
Formule 9
FORTRAN 34

G
Générateur de nombres 14, 15, 20
Géométrique (suite) 30
GOSUB 9
GOTO 9
Grammaire 32
Grégorien (calendrier) 12

H
Harmonique (suite) 30
Hasard 14
Heuristique 23
Histoire du calendrier 12

I
IF 9, 24
Infinie (série) 30
INPUT 9
Instruction 8
INT 13

J
Jeu de NIM 20
Jeu sur ordinateur 20
Jour de la semaine 12
Julien (calendrier) 13

L
Langages Basic 7, 34
LET 9
Lièvres 10
Limites 30, 31
LIST 11
Lister un programme 11
Longueur d'une chaîne 16

M
Magique (carré) 26
Manuel, mode d'emploi 7, 9
Matrice 27
Mémoire 6
Modèle 11, 22
Mot clé 8

N
NEXT 9
NEW 6
NIM (jeu de) 20
Nom 8
Nombre aléatoire 14, 20
Nombre entier 13
Nombre premier 25
Nom de variable 8
Numérique 8
Numéro de ligne 6

P
Paramètre 22
Pi 30, 31
Point décimal 8
Précision 30, 31
Premier (nombre) 25
PRINT 9
Programme trop grand 35
Programme trop lent 35

R
RANDOMIZE 32
Recherche 16, 18, 19
REM 9
Remarque dans un programme 9
Renard 10
RETURN 9
RND 15, 32

S
Saisie des données 9
Saut absolu 9
Saut conditionnel 9
Séries 30, 31
Signe égalité 9
Simulation 11
Sortie des résultats 9
Sous-programme 9
STOP 9
Stratégie 20
Suite alternée 31
Suite convergente 30
Suite divergente 30
Suite géométrique 30
Suite harmonique 30

T
Taux de natalité 28
Téléphonique (répertoire) 16
Temps de calcul 24
Tester 7
Tests (données pour) 6
TO 9
TRACE 7

V
Variable 8, 9

LES ÉDITIONS FOUCHER — 128, rue de Rivoli, 75001 Paris — N° 10358-286

Collection RENDEZ-VOUS MICRO

◇ **Première rencontre**
Guide d'initiation au micro-ordinateur

◇ **Pour mieux le comprendre**
Douze programmes expliqués. TO7 MO5

◇ **Voyage en BASIC**
Navette spatiale. MO5 AMSTRAD

◇ **A la découverte des bases de données**
Jeu d'aventure : labyrinthe et fonds marins
TO7 MO5 AMSTRAD MSX

◇ **Votre second à la maison**
Programmes pratiques, programmes de jeux
TO7 MO5 AMSTRAD COMMODORE

GUIDES PRATIQUES

- **SIL'Z 8 bits** - Modèles 2, 3, 4 et 8
- **TO7 (70) · MO5**
- **BULL · MICRAL** - 80-20/21/22 (G), 90-20
- **APPLE II** - Europlus e, c
- **IBM PC et compatibles**
  Persona, Micral 90-30, Goupil 4...

ISBN 2-216-00931-8

AMSTRAD CPC

MÉMOIRE ÉCRITE
MEMORY ENGRAVED
MEMORIA ESCRITA

https://acpc.me/