

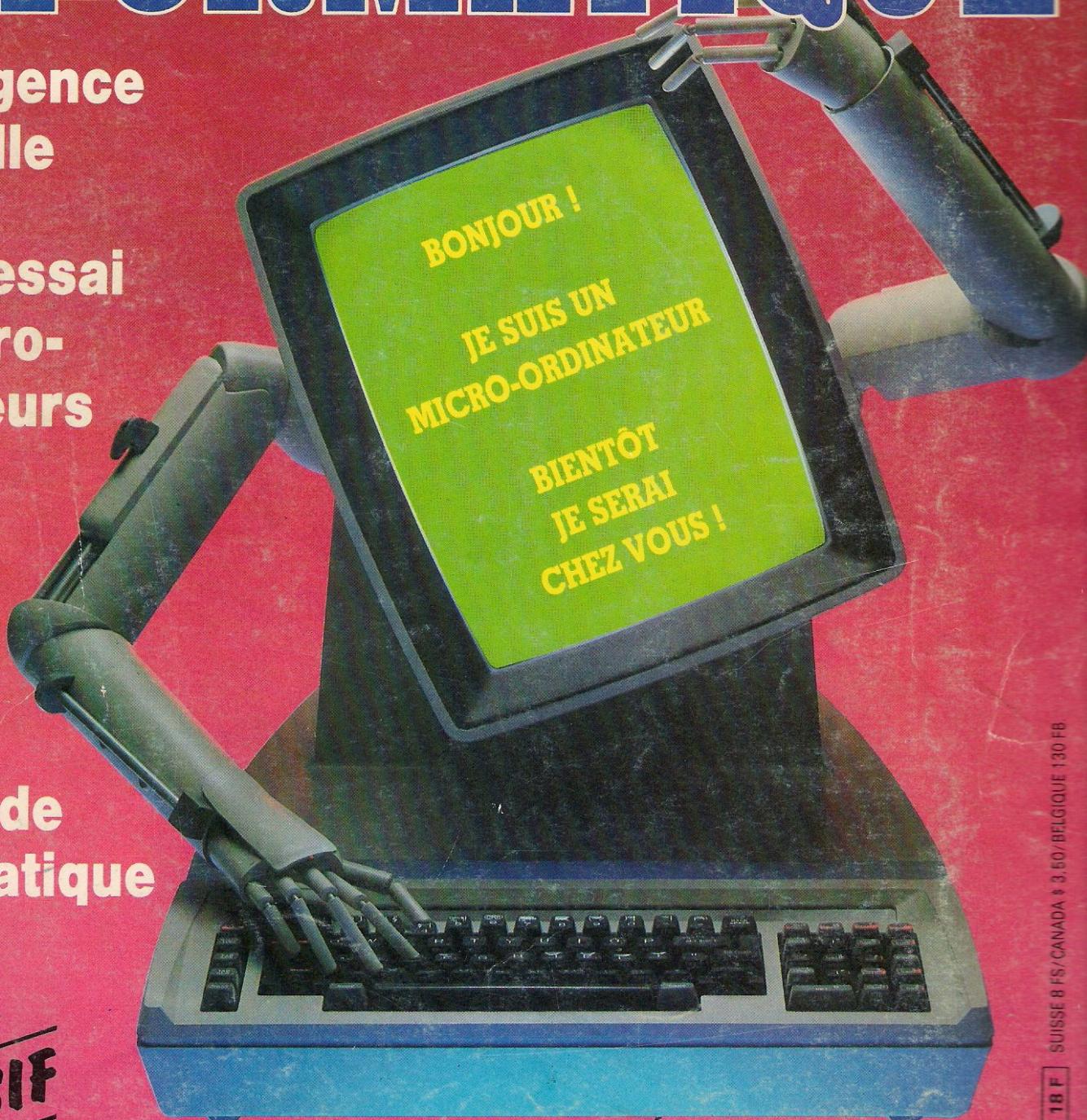
LA RÉVOLUTION INFORMATIQUE

L'intelligence artificielle

Banc d'essai des micro-ordinateurs

Les métiers de l'informatique

EXCLUSIF



DANS CE NUMÉRO
Ordinapoche
VOTRE PASSEPORT POUR L'INFORMATIQUE

Pour vous aider à comprendre l'informatique



voici

Ordinapoche

Mais l'informatique est encore chargée de mystère. Elle s'apparente aux mathématiques et souvent rebute nombre de ceux qui désireraient s'y lancer. Il faut démystifier l'informatique. Un ordinateur n'est qu'un outil comme un autre. Bien sûr, c'est un outil qui nous paraît « intelligent » et par conséquent susceptible de nous déposséder d'un des attributs de notre cerveau. C'est pourquoi comprendre le fonctionnement de l'ordinateur, les langages qu'il utilise, les relations entre son architecture interne et un programme, permet de pénétrer dans le monde de l'informatique et de se familiariser avec l'avenir qu'il nous prépare.

La révolution informatique s'est faite dans un laps de temps si court que l'on est assailli, agressé même par un vocabulaire nouveau. Au détour d'une publicité, d'un article, d'annonces ou d'émissions télévisées, on est exposé à des termes étranges : « cycle » d'un ordinateur, vitesses mesurées en « nanoseconde », capacité de mémoire de 16 K ou 32 K, « mémoire centrale », « auxiliaire », « ROM », « RAM ». Ces termes sont inquiétants parce que nouveaux, parce que non encore maîtrisés.

Une introduction simple à l'informatique doit permettre aujourd'hui à chacun de mieux comprendre et choisir les voies vers lesquelles nous engage cette nouvelle technologie. De dialoguer avec ceux qui s'en servent. De connaître les limites des ordinateurs. D'évaluer leurs capacités et leurs applications, lorsque nous avons à nous en servir au bureau, à l'usine, sur les chantiers et bientôt dans notre voiture ou à notre domicile. L'ordinateur se miniaturise. Comme la calculatrice, il va tenir dans le creux de la main, nous parler, écrire sur l'écran de notre téléviseur.

C'est la révolution informatique, l'ordinateur est partout. Le voici qui entre dans notre foyer.

Maxi, mini, micro, calcullettes. Sous toutes ces formes, c'est une nouvelle puissance de calcul pour chacun d'entre nous. Des « assistants intellectuels » capables d'écrire sur un écran de télévision, de nous parler avec une voix humaine, de jouer des notes de musique, de dessiner des graphiques en couleur. Les micro-ordinateurs ouvrent des horizons nouveaux et des applications à d'infinites variétés.

Nous sommes déjà familiarisés avec les ordinateurs capables de résERVER les places d'hôtel ou d'avion, de préparer les relevés de compte en banque, de gérer notre police d'assurance-vie, de calculer les salaires dans l'entre-

prise. Nous avons fait connaissance avec les ordinateurs des centres d'impôts, de la Sécurité sociale, ceux des usines ou des laboratoires, ou encore ceux qui assurent des services variés allant de la vente par correspondance aux annuaires et répertoires que l'on consultera directement de chez soi, grâce au couplage de la télévision et du téléphone. Aujourd'hui, non seulement, l'essor de la micro-informatique, mais aussi celui de la télématique, nous surprennent et nous questionnent.

Aussi nombreuses que puissent être les marques actuellement sur le marché, les ordinateurs ont la même structure de base. Tous utilisent des programmes écrits dans des langages, tantôt simples comme le Basic, tantôt plus complexes comme le Fortran, ou le Cobol.

Pour préparer le futur, pour vivre l'ère de l'informatique, voici ORDINAPOCHE.

Certes, ce n'est pas un ordinateur, ce n'est pas non plus une calculatrice, ce n'est même pas une règle à calcul. Il n'effectuera pas pour vous d'opérations mais, en s'associant à vous, il vous fera comprendre ce qu'est un ordinateur et ce qu'est un programme.

L'informatique nous concerne tous. Bien sûr on peut apprendre l'informatique en s'achetant un microprocesseur et en apprenant chez soi à programmer. Mais leur coût est encore très élevé pour un budget familial,

alors que **ORDINAPOCHE**, avec quelques tirettes en carton, vous permettra, en quelques heures, de comprendre la base informatique nécessaire pour affronter la révolution du futur.

ORDINAPOCHE est un nouveau type d'auxiliaire didactique, relevant d'une pédagogie systémique, c'est-à-dire tenant compte des inter-relations des éléments entre eux. Quand il s'agit de comprendre une technique aussi peu familière, au premier abord, que l'informatique, l'enseignement linéaire auquel nous avons été habitués (comprendre A pour passer à B, puis ensuite aborder C une fois

que l'on a compris) n'est pas approprié. Il faut entrer dans le mode de pensée de l'informatique par plusieurs portes à la fois. C'est à cela que **ORDINAPOCHE** vous prépare. Grâce à lui, vous vous plongez d'emblée dans l'informatique, vous simulez un ordinateur, le rôle d'un programme et progressivement vous en comprenez la véritable logique, la portée, la puissance et, partant, l'influence sur notre futur.

Voici **ORDINAPOCHE**, un simulateur en carton qui sera votre ami et votre guide pour vous faire entrer dans le monde prodigieux des ordinateurs.

Quelques termes utiles à connaître

(Vous les retrouverez évidemment au cours du texte)

Accumulateur : Organe de calcul d'un ordinateur.

Adresse : Endroit où réside en mémoire une instruction ou une donnée.

Assembleur : Programme permettant d'écrire des programmes en langage machine.

Bandes magnétiques : Mémoires auxiliaires permettant le stockage de quantités importantes d'informations.

Basic : Langage évolué créé en 1965 aux Etats-Unis, précis, mais surtout simple d'utilisation, en particulier pour les non-informaticiens.

Bit : Chiffre binaire d'information (zéro ou un).

Boucle : Séquence répétitive inscrite dans un programme grâce à une instruction spéciale.

Branchement inconditionnel : Instruction permettant un branchement obligatoire dans un programme.

Branchement conditionnel : Instruction permettant un branchement dans un programme, seulement si certaines conditions sont réunies.

Caractère : Un caractère représente 8 bits ou un octet.

Case de mémoire : Endroit identifié par une adresse, où sont rangées en mémoire les instructions et les données.

Code d'opération : Ordre codé que reçoit l'ordinateur pour exécuter une opération spécifique.

Compilateur : Programme permettant de traduire un langage évolué (Fortran, Basic) en langage Assembleur.

Compteur d'adresse : Compteur permettant de suivre une par une chaque instruction d'un programme.

Cycle d'ordinateur : Succession des étapes d'exécution d'un programme.

Décalage : Instruction permettant de décaler les chiffres dans l'accumulateur.

Décodeur d'instruction (dans Ordinapoch) : Fenêtre permettant la « traduction » en français des instructions d'un programme.

Disques magnétiques : Mémoires auxiliaires d'ordinateur et de micro-ordinateur.

Donnée : Information codée destinée à être traitée par l'ordinateur (par exemple : chiffres à additionner).

Edition : Programme spécial permettant d'écrire et de corriger un programme.

Entrée : Appareillage permettant d'introduire les données et les instructions dans l'ordinateur (clavier, écran, lecteur de cassettes ou de bandes).

Fortran : Langage évolué très utilisé en informatique (abréviation de FORmula TRANslator).

Instruction : Expression codée qui contient un ordre de traitement exécutable par l'ordinateur. Elle indique, notamment, la nature de l'opération à effectuer (code d'opération) et l'adresse des données à traiter.

K : $K = 2^{10} = 1\,024$ unités d'information.

Langage machine : Seul langage que comprend l'ordinateur, formé d'une succession de 0 et de 1.

Mémoire : Organe de l'ordinateur où sont rangées les instructions et les données, chaque « case mémoire » est identifiée par l'adresse.

Microseconde : Unité de temps utilisée en informatique (1 microseconde = 1 millième de seconde).

Mot et mot programme : Unité de stockage en mémoire, comprend le code d'opération et l'adresse formant une instruction (Ordinapoch utilise des mots de 3 chiffres décimaux).

Nanoseconde : Unité de temps utilisée en informatique (1 nanoseconde = 1 milliardième de seconde).

Octet : Un octet = 8 bits (zéro ou un).

Ordinogramme : Diagramme de déroulement de chaque étape des opérations devant être effectuées pour obtenir la solution à un problème particulier.

Programme : Suite ordonnée d'instructions formulées dans un langage approprié et précisant les différentes étapes du traitement informatique.

Programme de chargement : Programme spécial permettant d'introduire un programme par l'unité d'entrée.

Puce (dans Ordinapoch) : Indicateur en carton qui saute de case mémoire en case mémoire et joue le rôle de compteur d'adresse.

Registre d'instruction : Mémoire capable d'emmageriser chaque instruction au moment de son exécution.

Séquence d'appel : Instruction permettant d'appeler un sous-programme.

Sortie : Appareillage permettant de communiquer les résultats des traitements à l'extérieur (écran, imprimante...).

Sous-programme : Partie d'un programme mémorisé sur bande magnétique ou sur disque et pouvant être facilement utilisée par n'importe quel programme.

Temps réel : Traitement informatique sous forme interactive. Les résultats sont communiqués à l'utilisateur au fur et à mesure du traitement des données.

Test d'accumulateur (sur Ordinapoch) : Fenêtre permettant de vérifier le signe d'un nombre contenu dans l'accumulateur.

Unité de commande : Organe logique de contrôle du cycle de l'ordinateur. (Sur Ordinapoch : vous êtes l'unité de commande.)

Voici en 8 chapitres tout ce qu'il faut savoir pour tirer le meilleur parti de votre **Ordinapoche**

INTRODUCTION		
par Joël de Rosnay	21	
Quelques termes utiles à connaître	22	
Préambule	23	
1. Pour comprendre l'organisation d'un ordinateur, voici comment nous effectuons une addition	23	4. Que se passe-t-il à l'intérieur d'un ordinateur ? 28
2. Pour comprendre l'organisation d'un programme	25	5. Présentation complète d' <i>Ordinapoche</i> 30
3. Pour comprendre la logique d'un ordinateur	27	6. Comment écrire un programme avec <i>Ordinapoche</i> 32 Vos directives pas à pas pour vous servir d' <i>Ordinapoche</i> 32
		7. Deux aspects importants des programmes : boucles et branchements 35 les boucles 35 le branchement inconditionnel 36
		8. Pour comprendre la puissance et la souplesse de l'ordinateur 36 la prise de décision 36 la multiplication 37 le décalage de chiffres 38 les sous-programmes 39 Des programmes pour écrire des programmes 40
		ANNEXE Pour apprendre et concevoir un programme : le jeu de NIM à une rangée 41



Si vous êtes pressé ou si vous connaissez déjà l'informatique, allez page 32 et lisez le mode d'emploi rapide d'*Ordinapoche*.

Ce qu'est **Ordinapoche...** et ce qu'il n'est pas

Ordinapoche est un outil pédagogique pour comprendre l'informatique. Il illustre l'architecture et l'organisation de base de tous les ordinateurs. C'est presque un jeu. Il faut donc bien comprendre ce qu'il est capable de faire et quelles sont ses limites.

1. Ordinapoche ne fait pas les calculs. Il ne les facilite pas non plus.

2. Ordinapoche aide à comprendre

l'organisation interne, le fonctionnement d'un ordinateur et les techniques de programmation.

3. Ordinapoche est un simulateur. Il démonte les opérations de base. C'est vous qui serez la source d'énergie. C'est vous qui serez l'unité de commande.

Pour faciliter vos premiers contacts avec l'informatique, vous pouvez procéder de la manière suivante :

1. Si vous êtes très pressé, lisez le descriptif d'utilisation d'*Ordinapoche* en page 32.

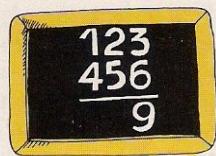
2. Pour ceux qui souhaitent comprendre étape par étape l'architecture de l'ordinateur, la logique de la programmation et écrire eux-mêmes leur premier programme, lisez tranquillement le présent livret d'accompagnement. Le voyage en vaut la peine.

1 Pour comprendre l'organisation d'un ordinateur voici comment nous effectuons une addition.

Avant de s'aventurer dans la description des ordinateurs ou d'*Ordinapoche*, il est utile de savoir que l'ordinateur effectue une addition un peu comme nous la faisons nous-mêmes. Il est donc nécessaire de détailler ce que comporte un calcul très simple.

Reportez-vous au temps de vos premières leçons d'arithmétique. Vous êtes de retour à l'école, devant le tableau noir, et l'institutrice vous demande d'additionner deux nombres : par exemple 123 et 456. Que faites-vous ? Vous inscrivez d'abord les deux nombres au tableau, en les

disposant l'un sous l'autre en colonne, puis vous tracez une ligne sous ces nombres et additionnez à partir de la colonne de droite jusqu'à la colonne de gauche (3 et 6 font 9 ; 2 et 5 font 7 ; 1 et 4 font 5). Le résultat est 579. Est-ce fini ? Pas tout à fait. Vous savez que l'institutrice attend le



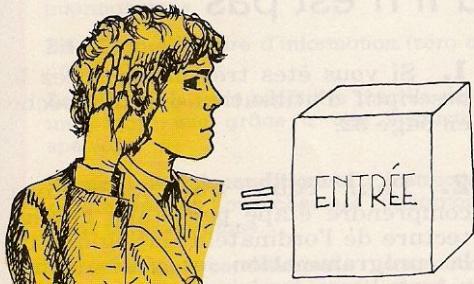
résultat et vous annoncez : la réponse est 579. Maintenant votre addition est complète et terminée.

● Schéma simplifié d'un ordinateur.

Voyons maintenant ce que vous avez fait pour obtenir le résultat. Nous pourrons en même temps établir le premier schéma, simplifié certes, mais fonctionnel, d'un ordinateur capable d'effectuer les mêmes opérations que vous.

● Entrée.

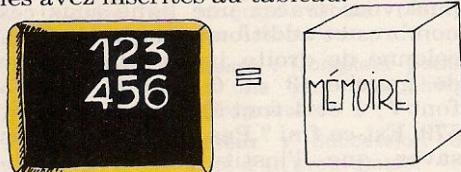
En premier lieu vous avez « écouté ». Dès que l'institutrice a mentionné votre nom, vous avez pris des notes mentales de tout ce qu'elle disait, portant une attention particulière à trois mots. Ces mots étaient « additionner » (c'est une *instruction*) ; 123 (c'est une *donnée*) et 456 (une autre donnée).



Ordinapoche (comme tout ordinateur) devra aussi posséder un dispositif « d'entrée » capable d'« écouter », c'est-à-dire de recevoir les instructions et les données.

● Mémoire.

On vous a remis des données, c'est-à-dire les deux nombres à additionner. Il vous fallait un dispositif convenable pour les conserver et les garder en mémoire. Pour cela vous les avez inscrites au tableau.

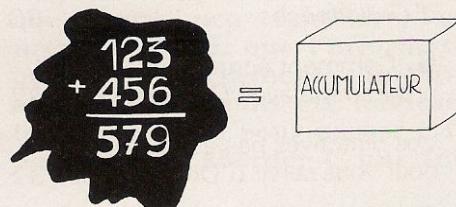


Dessins d'Anne PUYBARAUD

Ordinapoche (comme tout ordinateur) aura lui aussi besoin d'une mémoire.

● Accumulateur.

Vous avez ensuite additionné les deux nombres, c'est-à-dire les données, colonne par colonne, écrivant chaque résultat partiel dans la colonne correspondante jusqu'à ce que vous ayez obtenu le total que vous avez écrit sous le trait horizontal. Vous avez donc exécuté l'opération de calcul requise.



Ordinapoche (comme tout ordinateur) doit posséder lui aussi un organe de calcul. Etant donné que les résultats de chaque opération vont s'accumuler à l'intérieur de cet organe, nous l'appellerons « l'accumulateur ». On peut comparer cet accumulateur à un « boulier » dans lequel les boules seraient remplacées par des impulsions électroniques.

● Programme.

Souvenez-vous que lorsque vous avez enregistré vos données au

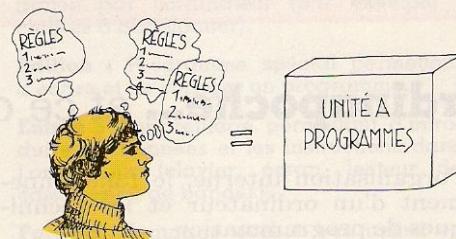


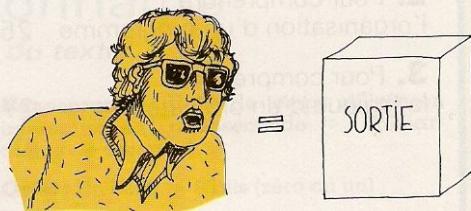
tableau (mémoire), vous ne les avez pas simplement placées au hasard. Au contraire, vous les avez inscrites avec

soin l'une sous l'autre, en colonne suivant la méthode que vous aviez apprise pour réaliser une addition. Souvenez-vous aussi que vous avez additionné ces nombres en commençant par établir la somme des chiffres de la colonne de droite, puis de la colonne suivante, et ainsi de suite. Chaque étape a donc été effectuée selon une *série de règles* bien définies que l'on vous avait enseignées auparavant. Ces règles qui vous ont guidé pour trouver la solution au problème portent le nom de « programme ».

Ordinapoche (comme tout ordinateur) ne serait d'aucune utilité sans programme.

● Sortie.

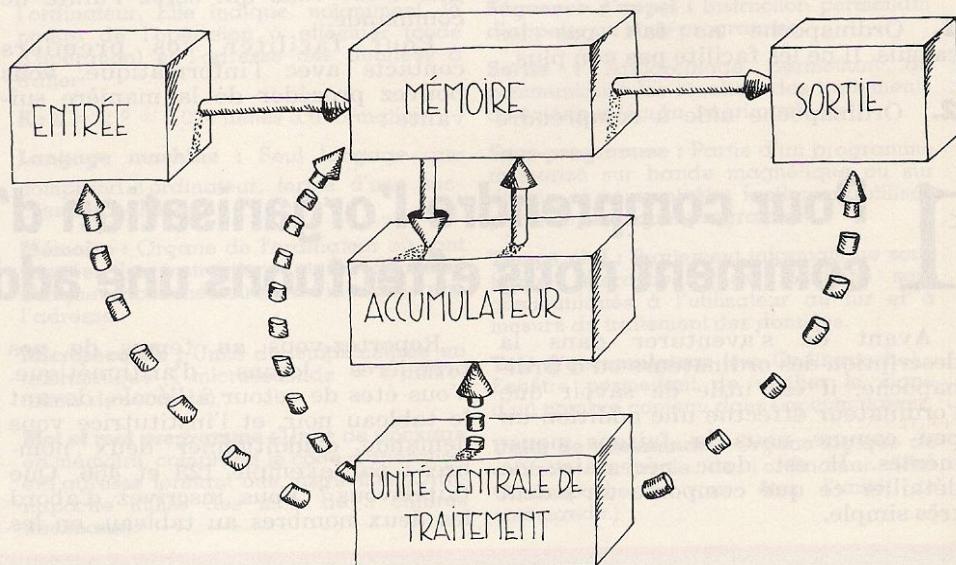
L'acte que vous avez accompli avant d'effacer le tableau et de retourner à votre place, fut de donner la réponse à l'institutrice.



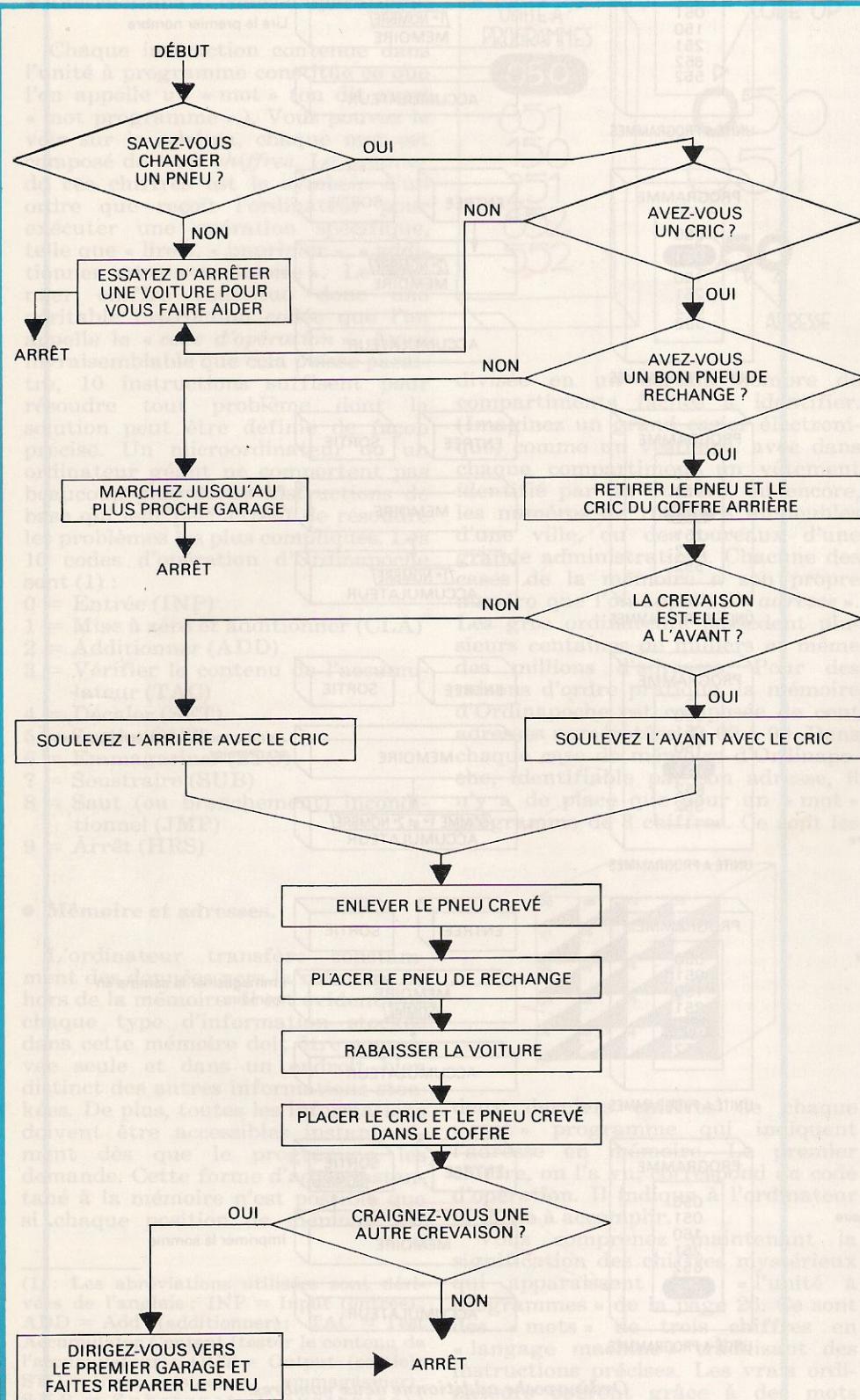
Ordinapoche (comme tout ordinateur) devra aussi pouvoir mettre ses réponses à notre disposition sous une forme facile à reconnaître. Pour ce faire, il lui faudra un dispositif de « sortie ». C'est un peu la « parole » de la machine.

Nous avons maintenant les éléments de base pour dessiner le premier schéma simplifié d'un ordinateur et d'Ordinapoche, comportant les différents éléments que nous venons de passer en revue : « entrée », « mémoire », « accumulateur », « programme » et « sortie ».

C'est le schéma de base de tout ordinateur, qu'il s'agisse d'un micro-ordinateur ou d'une IBM 4341.



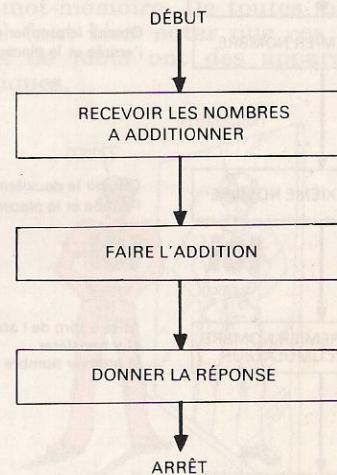
2 Pour comprendre l'organisation d'un programme.



Le schéma de la page 24 illustre les organes de base d'un ordinateur simple. Toutefois, il nous renseigne assez peu sur les interactions entre ces différents organes. Le programme sert justement à les relier les uns aux autres, étape par étape.

Une façon utile et très simple de se faire une première idée du mécanisme interne d'un ordinateur est de tracer ce que l'on appelle un « ordinogramme » ; c'est un diagramme du déroulement de chaque étape des opérations devant être effectuées pour obtenir la solution à un problème particulier. On peut tracer des ordinogrammes pour toutes sortes d'activités imaginables, même pour la réparation d'une crevaison comme l'indique l'ordinogramme ci-dessous.

L'ordinogramme ci-dessous illustre, lui, les étapes d'une addition de deux nombres.



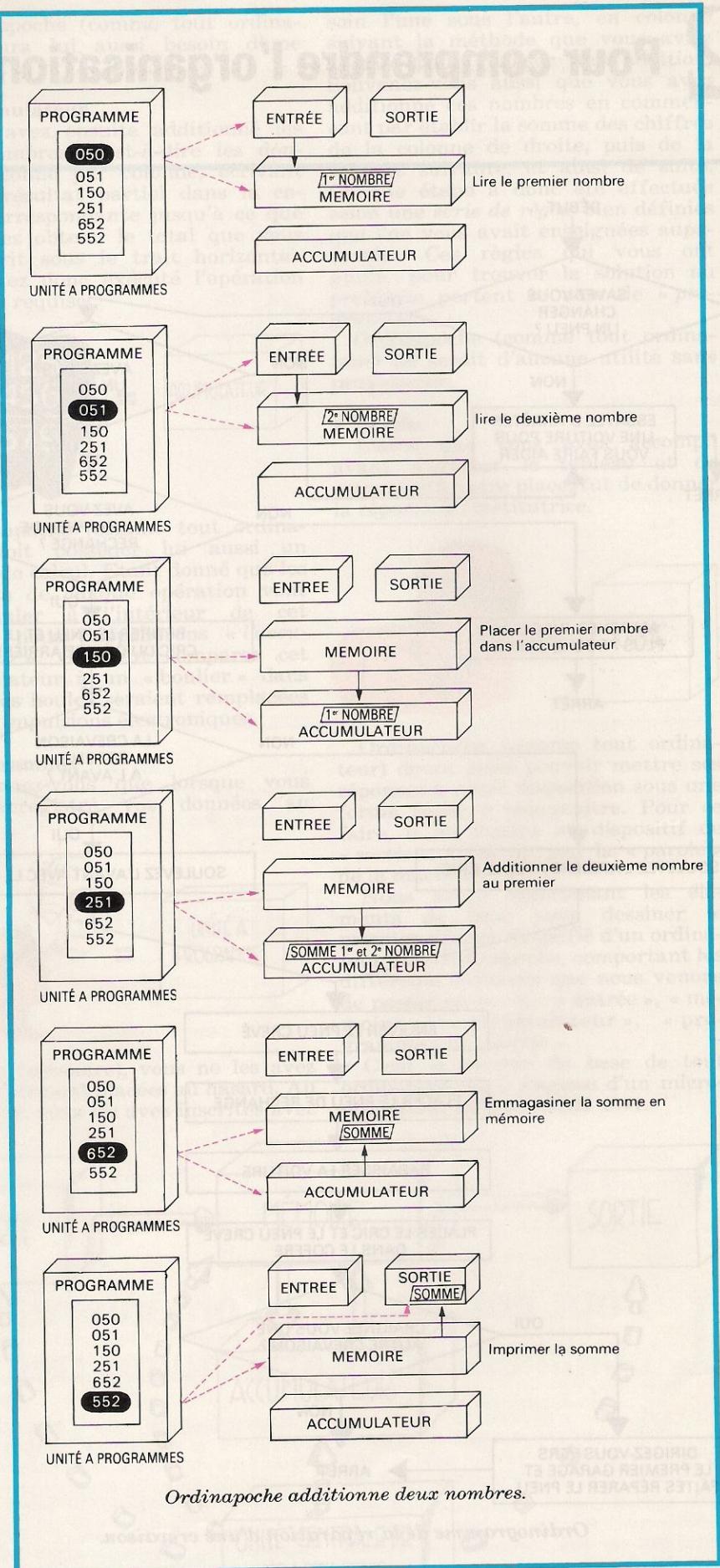
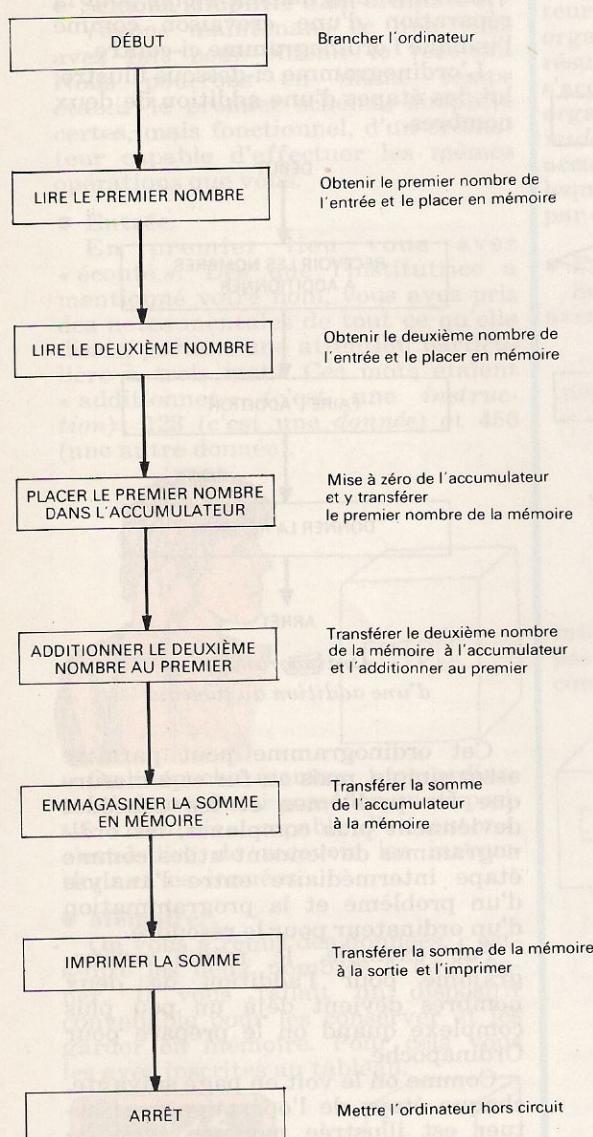
Ordinogramme d'une addition au tableau.

Cet ordinogramme peut paraître assez simple, mais au fur et à mesure que les problèmes et les procédés deviennent plus complexes, les ordinogrammes deviennent utiles comme étape intermédiaire entre l'analyse d'un problème et la programmation d'un ordinateur pour le résoudre.

Par exemple, le même ordinogramme pour l'addition de deux nombres devient déjà un peu plus complexe quand on le prépare pour Ordinapoché.

Comme on le voit en page suivante, chaque étape de l'opération à effectuer est illustrée par une série de cases. Le cheminement des données est représenté par des traits pleins, les lignes pointillées indiquent le cheminement des impulsions électro-

niques de contrôle. Sur la partie gauche du schéma on voit figurer une case intitulée pour le moment « unité à programmes » dans laquelle figurent des nombres d'apparence mystérieuse qui sont des abréviations en langage machine des instructions verbales contenues dans l'ordinogramme. Leur signification deviendra plus claire au cours des pages suivantes. Pour l'instant, il est suffisant de savoir que l'ordinateur les comprend parfaitement et que l'unité à programme les lit dans la bonne séquence.



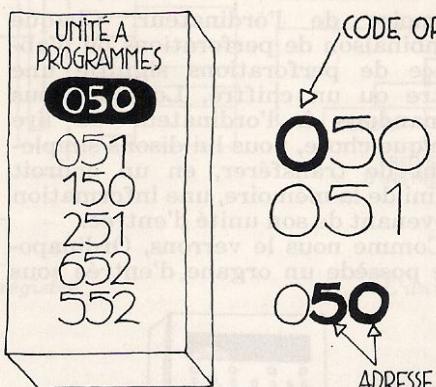
Ordinogramme de l'addition avec ordinapoch

3 Pour comprendre la logique d'un ordinateur.

● Instructions et codes d'opérations.

Chaque instruction contenue dans l'unité à programme constitue ce que l'on appelle un « mot » (on dit aussi « mot programme »). Vous pouvez le voir sur le schéma, chaque mot est composé de *trois chiffres*. Le premier de ces chiffres est le symbole d'un ordre que reçoit l'ordinateur pour exécuter une opération spécifique, telle que « lire », « imprimer », « additionner » ou « soustraire ». Le premier chiffre constitue donc une véritable instruction codée que l'on appelle le « *code d'opération* ». Aussi invraisemblable que cela puisse paraître, 10 instructions suffisent pour résoudre tout problème dont la solution peut être définie de façon précise. Un microordinateur ou un ordinateur géant ne comportent pas beaucoup plus de 10 instructions de base qui leur permettent de résoudre les problèmes les plus compliqués. Les 10 codes d'opération d'Ordinapoche sont (1) :

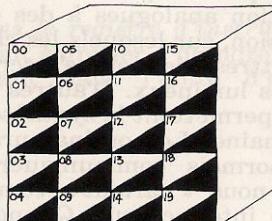
- 0 = Entrée (INP)
- 1 = Mise à zéro et additionner (CLA)
- 2 = Additionner (ADD)
- 3 = Vérifier le contenu de l'accumulateur (TAC)
- 4 = Décaler (SFT)
- 5 = Sortie (OUT)
- 6 = Emmagasiner (STO)
- 7 = Soustraire (SUB)
- 8 = Saut (ou branchement) inconditionnel (JMP)
- 9 = Arrêt (HRS)



divisée en un certain nombre de compartiments faciles à identifier. (Imaginez un grand casier électronique, comme un vestiaire, avec dans chaque compartiment un vêtement identifié par un numéro, ou encore, les numéros de rues des immeubles d'une ville, ou des bureaux d'une grande administration). Chacune des cases de la mémoire a son propre numéro que l'on appelle l'*« adresse »*. Les gros ordinateurs possèdent plusieurs centaines de milliers et même des millions d'adresses. Pour des raisons d'ordre pratique, la mémoire d'Ordinapoche est composée de cent adresses numérotées de 00 à 99. Dans chaque case de mémoire d'Ordinapoche, identifiable par son adresse, il n'y a de place que pour un « mot » programme de 3 chiffres. Ce sont les

● Mémoire et adresses.

L'ordinateur transfère constamment des données vers la mémoire ou hors de la mémoire. Il est évident que chaque type d'information stockée dans cette mémoire doit être conservée seule et dans un endroit bien distinct des autres informations stockées. De plus, toutes les informations doivent être accessibles instantanément dès que le programme les demande. Cette forme d'accès instantané à la mémoire n'est possible que si chaque position de mémoire est



deux derniers chiffres de chaque « mot » programme qui indiquent l'adresse en mémoire. Le premier chiffre, on l'a vu, correspond au code d'opération. Il indique à l'ordinateur la tâche à accomplir.

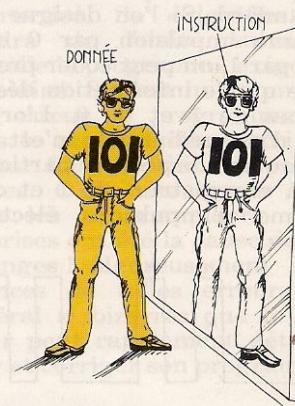
Vous comprenez maintenant la signification des chiffres mystérieux qui apparaissent dans « l'unité à programmes » de la page 26. Ce sont des « mots » de trois chiffres en « langage machine » traduisant des instructions précises. Les vrais ordinateurs opèrent grâce à des mots beaucoup plus longs que ceux d'Ordinapoche. Ils ont aussi beaucoup plus d'adresses en mémoire, mais le prin-

(1) : Les abréviations utilisées sont dérivées de l'anglais : INP = Input (entrée); ADD = Add (additionner); TAC = Test Accumulator Content (tester le contenu de l'accumulateur); OUT = Output (sortie); STO = Store (stocker, emmagasiner); SUB = Subtract (soustraire); JMP = Jump (aller à, ou, sauter à); HRS = Halt and Reset (arrêt et remise à zéro).

cipe de base est exactement le même : le début de chaque mot indique à l'ordinateur ce qu'il doit faire, et l'autre partie, où il doit aller ranger ou chercher dans la mémoire les données nécessaires à ses opérations.

● Les instructions et les données sont des mots qui se ressemblent.

Ce qui est remarquable, c'est que toutes les instructions peuvent être représentées par un code de 3 chiffres. Mais si les *instructions* sont représentées par des mots de 3 chiffres, comment les *données* sont-elles représentées ? Eh bien, de la même façon ! Les données, (c'est-à-dire la matière à traiter par l'ordinateur) sont également représentées par 3 chiffres. Plus précisément, leur taille est limitée à 3 chiffres, étant donné que c'est la capacité maximum d'un mot-mémoire. De toutes manières, il faut bien noter que ces deux sortes de mots ont des apparences identiques.



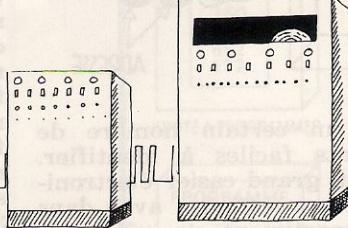
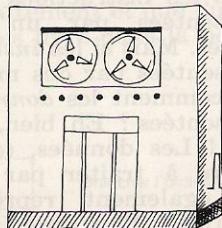
Cette ressemblance présente un grand avantage. Tout d'abord les deux sortes de mots peuvent être traitées par les mêmes équipements. Ils peuvent être introduits dans les mêmes unités d'entrée, être traités dans le même accumulateur, et être stockés dans la même mémoire. Cela ne représente pas seulement une grande économie, mais permet aussi à un ordinateur en fonctionnement, de modifier ses propres instructions. Les ordinateurs qui fonctionnent de cette manière, sont appelés *ordinateurs à programme enregistré*. Cette facilité d'emmagasiner et de modifier leur propre programme, donne à ces ordinateurs l'apparence d'être complètement automatiques. On verra plus loin comment l'ordinateur distingue les données des instruc-

4 Que se passe-t-il à l'intérieur d'un ordinateur ?

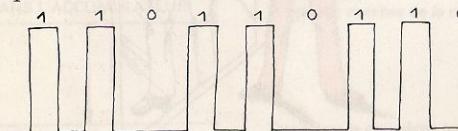
● Unité d'entrée (Input) :

Il y a deux instructions auxquelles les utilisateurs des ordinateurs accordent une signification particulière. Ces deux instructions sont « LIRE » (READ) et « IMPRIMER » (PRINT). Leur emploi implique l'utilisation des unités d'entrée et des unités de sortie de l'ordinateur.

Les ordinateurs n'étant que des machines, leurs différents organes



communiquent les uns avec les autres dans un langage qui leur est propre : un langage électronique. L'alphabet de ce langage consiste en impulsions électroniques et les mots, en séquences d'impulsions disposées selon des codes standard. Si l'on désigne l'absence d'une impulsion par 0 et sa présence par 1, on peut coder presque n'importe quelle information désirée, en groupes de 1 et de 0. L'organe d'entrée d'un ordinateur n'est rien d'autre qu'un dispositif particulier servant à enregistrer ces 1 et ces 0 sous forme d'impulsions électroniques.

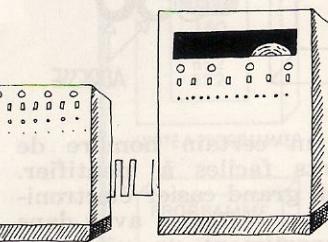


Parmi les dispositifs d'entrée les plus utilisés, figurent bien entendu les cartes perforées et les claviers, mais aussi les bandes magnétiques de grande taille ou miniaturisées sous forme de cassettes (comme celles que l'on utilise dans les magnétophones à cassettes), des disques magnétiques souples (minidisques, disquettes) ou des disques de grandes capacités servant à stocker des quantités très importantes d'informations.

Sur la carte ou sur la bande perforée les 0 et les 1 sont codés sous forme de trous ou sous forme de pleins. Le lecteur de cartes ou le lecteur de bandes est muni de palpeurs qui repèrent la présence ou l'absence de perforations dans les cartes ou dans les bandes. Ces brosses agissent comme des commutateurs qui transmettent les impulsions (ou signalent leur absence) à l'unité de

mémoire de l'ordinateur. Chaque combinaison de perforations ou d'absence de perforations signifie une lettre ou un chiffre. Lorsque nous demandons à l'ordinateur de lire quelque chose, nous lui disons simplement de transférer, en un endroit défini de la mémoire, une information provenant de son unité d'entrée.

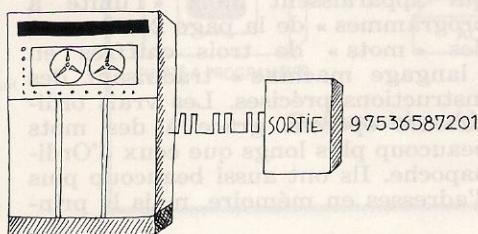
Comme nous le verrons, Ordinapoché possède un organe d'entrée sous



la forme d'un ruban de carton sur lequel on peut écrire des chiffres, et les effacer très facilement.

● Unité de sortie (Output)

Les organes de sortie permettent de transformer les impulsions stockées dans la mémoire en un langage que nous pouvons comprendre. Il fonctionne en quelque sorte à l'inverse des organes d'entrée. Parmi les systèmes les plus répandus figurent les imprimantes à grande vitesse capables d'imprimer des informations reçues de la mémoire à la vitesse de centaines de lignes à la minute. On utilise de plus en plus des écrans de visualisation analogues à des écrans de télévision, sur lesquels les chiffres ou les lettres viennent s'inscrire en caractères lumineux. D'autres unités de sortie permettent la synthèse de la voix humaine. Les ordinateurs peuvent désormais communiquer avec nous en nous « parlant » comme le ferait un interlocuteur. Quand nous demandons donc à un ordinateur d'imprimer quelque chose, nous lui disons seulement de retrouver une certaine information dans un endroit spécifique de la mémoire et de l'imprimer, soit sur du papier, soit de l'afficher sur un écran de télévision, soit encore de transformer ces informations en phrases que notre oreille peut entendre.



● Que se passe-t-il à l'intérieur d'un ordinateur ?

La série de schémas qui va suivre permettra de comprendre ce qui se passe effectivement à l'intérieur d'un ordinateur. Ces schémas reprennent les organes qui constituent un ordinateur simplifié et que l'on retrouve dans ordinapoché. Mais « l'unité à programmes » qui avait été utilisée précédemment pour présenter la logique interne des programmes se subdivise maintenant en *trois éléments essentiels* : le *registre d'instruction*, le *compteur d'adresses*, et l'*unité de commande*.

● Le registre d'instruction

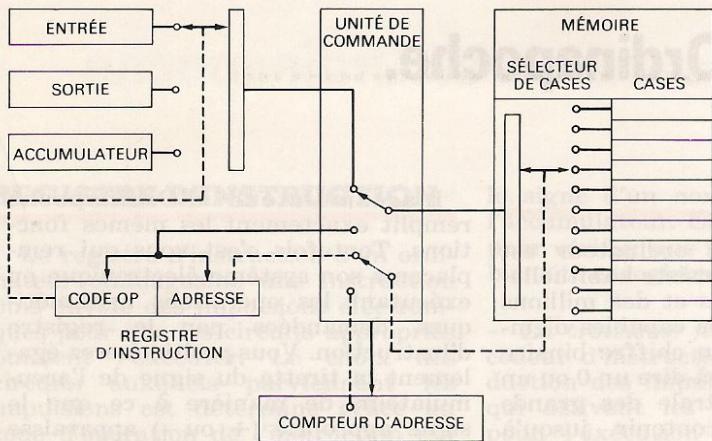
Le rôle du registre d'instruction est d'emmageriser chaque « mot-programme » ou instruction, au moment même de son exécution. Une fois l'exécution de cette instruction terminée, le registre est prêt à recevoir une nouvelle instruction.

● Le compteur d'adresses

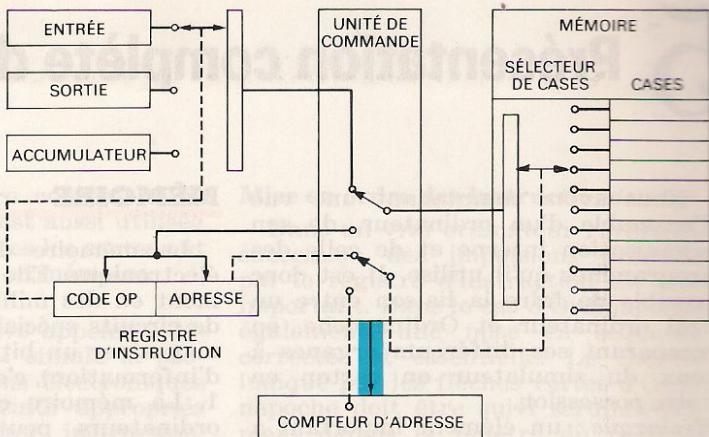
Quand vous avez à vous concentrer sur une longue liste de mots ou de chiffres, vous placez souvent votre doigt sur la première ligne et vous suivez de ligne en ligne afin d'être certain de lire la bonne ligne. Un compteur d'adresses ne fait pas autrement. Il contient l'adresse de la case de la mémoire d'où l'instruction en cours vient d'être retirée. Avant d'exécuter l'instruction il suffit d'ajouter 1 à cette adresse pour obtenir automatiquement l'adresse exacte de l'instruction suivante, puisque les instructions sont emmagasinées les unes derrière les autres dans la mémoire. Le compteur d'adresses a d'autres propriétés, nous les verrons plus tard lorsque nous rencontrons pour la première fois des instructions de « saut » permettant des branchements dans le programme ou permettant d'accéder à d'autres adresses qui ne se suivent pas en séquences les unes avec les autres.

● L'unité de commande

L'unité de commande contrôle le travail du registre d'instructions et du compteur d'adresses en accord avec l'ensemble de tous les autres organes de l'ordinateur. C'est un dispositif qui agit un peu comme le fait une téléphoniste, établissant les communications quand les signaux lumineux apparaissent sur son standard. Plus précisément l'unité de



Ordinateur à programme enregistré.



1. – L'unité de commande ajoute 1 au compteur d'adresse.

commande est capable d'effectuer les fonctions suivantes :

1° Ajouter 1 au nombre contenu dans le compteur d'adresse, ce qui donne automatiquement l'adresse de l'instruction suivante.

2° Utiliser le nombre contenu dans le compteur d'adresse pour repérer l'adresse de la prochaine instruction et pour la communiquer au registre d'instructions. Le compteur d'adresse dirige le sélecteur de cases vers la case appropriée.

3° Commander au registre d'instruction d'exécuter l'instruction en cours. Dans le cas présent l'instruction exécutée permet de transmettre une donnée de l'entrée à la mémoire.

Les schémas 1, 2, 3 indiquent en couleur les connexions internes qui se font dans l'ordinateur sous forme d'impulsions codées. Ces impulsions électroniques circulent dans des circuits logiques qui permettent l'ouverture ou la fermeture « d'interrupteurs » et ainsi l'aiguillage des infor-

mations vers les organes ou les centres appropriés.

● Comment l'ordinateur peut-il différencier une instruction d'une donnée ?

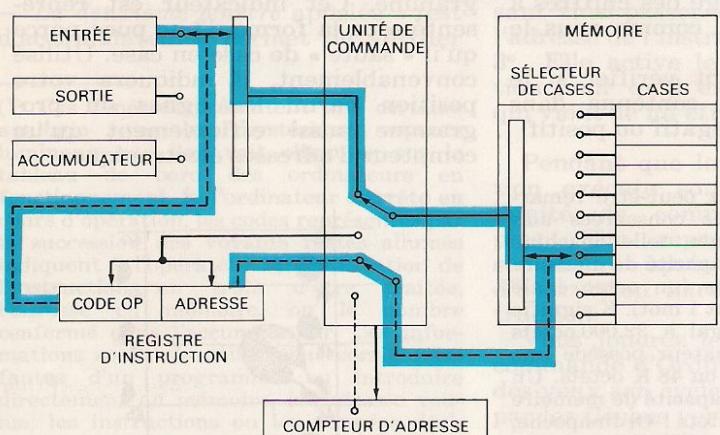
Dans un ordinateur l'utilisation et la signification d'un mot dépendent strictement de l'unité où ce dernier se trouve. A titre de comparaison, la suite de nombres 95-60-90 par exemple, peut s'interpréter différemment selon l'utilisation qu'on leur donne. Dans un studio de cinéma, ils pourraient indiquer les mensurations d'une actrice aux lignes avantageuses (données). Dans une chambre forte, le chiffre permettant d'ouvrir un coffre-fort (instructions). Dans un cours d'arithmétique élémentaire, ce pourrait être tout simplement trois nombres à additionner (données). Par exemple :

● Dans le registre d'instruction, 017 pourrait signifier « prendre le mot se trouvant dans l'unité d'entrée et

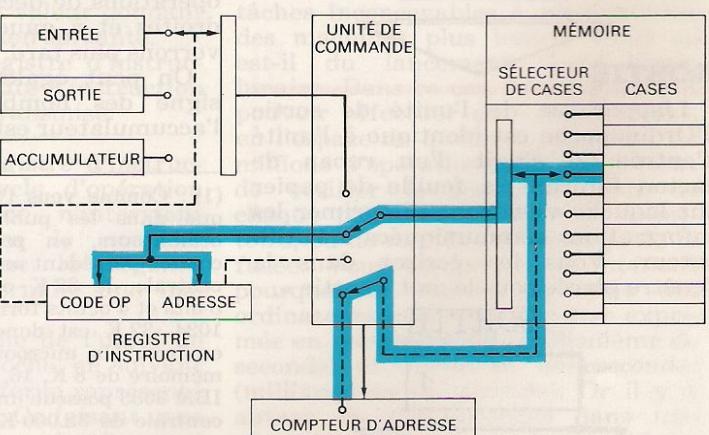
l'emmager dans la case mémoire n° 17 ».

● Dans l'accumulateur, ce mot sera simplement considérée comme une donnée représentant le nombre 17, et il sera additionné ou soustrait à tout autre nombre se trouvant déjà dans l'accumulateur.

● Dans la mémoire, 017 peut représenter soit une donnée soit une instruction. Sa « signification » dépendra de l'unité suivante vers laquelle le programme exige qu'il soit acheminé. Si c'est une donnée et que par erreur le programme l'appelle dans le registre d'instruction, l'ordinateur fournira des réponses incohérentes. C'est une erreur fréquente chez les programmeurs. De telles méprises ont été la cause de résultats étranges ! Heureusement les conséquences de telles erreurs sont en général si bizarres que le programmeur peut rapidement détecter l'erreur et corriger son programme.



2. – Le registre d'instruction exécutant une lecture.

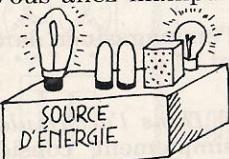


3. – L'unité de commande communique un mot au registre d'instruction.

5 Présentation complète d'Ordinapoche.

Nous avons maintenant une vue d'ensemble d'un ordinateur, de son organisation interne et de celle des programmes qu'il utilise. Il est donc possible de faire la liaison entre un vrai ordinateur et Ordinapoche, en comparant ses différents organes à ceux du simulateur en carton en votre possession.

Remarque : un élément important a été volontairement omis dans Ordinapoche. C'est la source d'énergie. Avec Ordinapoche c'est *vous qui êtes* la source d'énergie. Vous allez manipuler



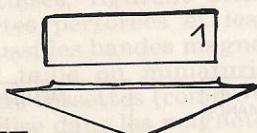
VOUS =

les tirettes et déplacer les données d'une section à une autre. Vous effectuerez également les opérations arithmétiques qui doivent être exécutées dans l'accumulateur. Ceci n'enlève rien à Ordinapoche en tant qu'auxiliaire didactique. Rappelez-vous que vous n'utilisez pas Ordinapoche pour apprendre à faire des opérations arithmétiques, mais pour comprendre le fonctionnement interne d'un ordinateur.

ENTRÉE

L'unité d'entrée d'Ordinapoche est constituée par un ruban en carton figurant soit un paquet de cartes perforées, soit une bande perforée. On fait apparaître dans la fenêtre placée sous le mot « entrée » la carte numéro 1, puis la carte numéro 2 selon l'exécution des programmes, comme on le verra.

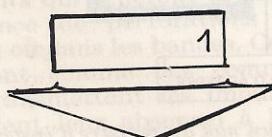
ENTRÉE



SORTIE

L'apparence de l'unité de sortie d'Ordinapoche est identique à l'unité d'entrée. Il s'agit d'un ruban de papier sur laquelle viendront s'imprimer les informations communiquées à l'utilisateur. Vous les écrirez dans la fenêtre placée sous le mot « sortie ».

SORTIE



MÉMOIRE

La mémoire d'un ordinateur est électronique. Elle consiste habituellement en des millions et des millions de circuits spécialisés capables d'emmager un bit (ou chiffre binaire d'information), c'est-à-dire un 0 ou un 1. La mémoire centrale des grands ordinateurs peut contenir jusqu'à 32 millions de caractères (chaque caractère représente 8 bits. Soit un octet). Sur le plan pratique on parle de « mots » de 36 chiffres binaires ou plus. Pour plus de simplicité, Ordinapoche utilise le système décimal plutôt que le système binaire. Sa mémoire est beaucoup plus petite, elle ne peut contenir que 100 mots de 3 chiffres. Ceux-ci sont écrits au crayon ou au stylo feutre effaçable et peuvent être consultés visuellement (1).

CASE N°	CONTENU	CASE N°	CONTENU	CASE N°
00	00	20	00	40
01	01	21	01	41

Etant donné que la mémoire d'Ordinapoche ne peut emmagasiner que des nombres de 3 chiffres, vous pouvez être surpris de l'inclusion d'un carré supplémentaire dans la rangée inférieure de l'accumulateur. Ce carré est prévu pour contenir le « dépassement » de capacité qui résulte de l'addition de deux nombres de 3 chiffres dont la somme est supérieure à 999.

ACCUMULATEUR

L'accumulateur est l'unité arithmétique de l'ordinateur. C'est dans celle-ci que se font les opérations d'addition, de soustraction et aussi les opérations de décalage des chiffres à droites et à gauche, comme nous le verrons plus tard.

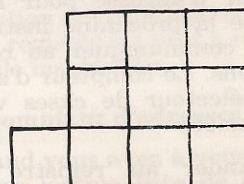
On peut également vérifier si le signe des nombres contenus dans l'accumulateur est négatif ou positif.

(1) : Comme vous l'avez peut-être remarqué dans les publicités consacrées aux ordinateurs, on présente telle machine comme possédant une capacité de mémoire centrale de 32 K octets (un octet égale 8 bits et 4 octets forment 1 mot). K signifie 1024, 32 K est donc égal à 32.000 octets environ. Un microordinateur possède une mémoire de 8 K, 16, 32 ou 48 K octets. Un IBM 3033 possède une capacité de mémoire centrale de 32.000 K octets ! Ordinapoche, quant à lui, n'a que 0,3 K de mémoire centrale.

L'accumulateur d'Ordinapoche remplit exactement les mêmes fonctions. Toutefois c'est vous qui remplacez son système électronique en exécutant les opérations arithmétiques demandées par le registre d'instruction. Vous positionnerez également la tirette du signe de l'accumulateur, de manière à ce que le signe approprié (+ ou -) apparaisse bien dans la fenêtre circulaire.

Au moment d'introduire les données sur les cartes d'entrée ou de sortie, ou dans la mémoire, n'oubliez pas de préciser si elles sont négatives. Les nombres sans signes sont considérés comme positifs.

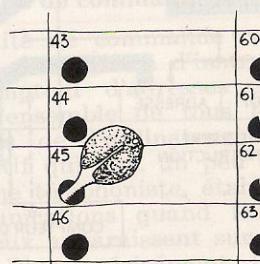
L'accumulateur proprement dit se compose simplement de la rangée inférieure des carrés que voyez ci-contre. Les deux rangées supérieures servant de bloc-note ou de brouillon pour l'addition et la soustraction.



ACCUMULATEUR

COMPTEUR D'ADRESSES

Le compteur d'adresses mémorise la position de la prochaine ligne du programme à exécuter. C'est en fait un compteur électronique dont le contenu représente l'adresse de la case-mémoire où doit être cherchée l'instruction suivante. Le compteur-d'adresses d'Ordinapoche est un indicateur en carton en forme de « puce » qui se place manuellement d'une case à une autre dans les trous prévus à cet effet durant l'exécution du programme. Cet indicateur est représenté sous la forme d'une puce parce qu'il « saute » de case en case. Utilisé convenablement, il indiquera votre position parmi les lignes du programme aussi efficacement qu'un compteur d'adresses électronique.



REGISTRE D'INSTRUCTION

Le registre d'instruction d'un ordinateur emmagasine une instruction puis envoie des impulsions électroniques pour que les circuits appropriés puissent l'exécuter. Le choix des circuits auxquels parviennent les impulsions est déterminé grâce au code d'opération de l'instruction qui se trouve dans le registre instruction. C'est exactement ce qui se produit lorsque vous composez au cadran de votre téléphone les huit chiffres d'un numéro interurbain. Les deux premiers chiffres servent d'indicatif régional et activent l'équipement qui dirige l'appel vers la zone désirée. Les deux chiffres suivant sélectionnent le central approprié dans cette zone et les quatre derniers chiffres agissent sur l'équipement de commutation nécessaire pour vous mettre en liaison avec l'appareil de votre correspondant. Le registre d'instruction d'Ordinapoch se compose de tirettes en carton donnant le code d'opération et l'adresse. Trois fenêtres de visualisation permettent d'afficher les informations imprimées sur chaque tirette (1).



— La première fenêtre appelée « registre d'instruction » vous permet de regarder à « l'intérieur » du registre pour voir l'instruction qui s'y trouve emmagasinée.

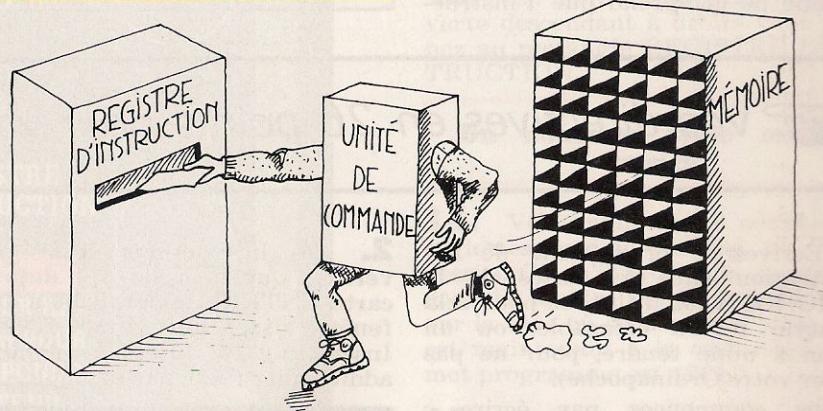
— La deuxième fenêtre appelée « test d'accumulateur » permet de vérifier

(1) Ces fenêtres présentent une certaine analogie avec les mystérieux voyants lumineux que l'on voit clignoter sur le tableau de bord des ordinateurs en fonctionnement. Si l'ordinateur s'arrête en cours d'opération, les codes représentés par la succession des voyants restés allumés indiquent à l'opérateur la signification de l'instruction en train d'être traitée, l'adresse en mémoire, ou le nombre renfermé dans l'accumulateur. Ces informations sont très utiles pour corriger les fautes d'un programme ou introduire directement en mémoire, à l'adresse voulue, les instructions ou les données désirées, en agissant sur les touches du tableau de bord.

le signe d'un nombre contenu dans l'accumulateur. Elle est aussi utilisée pour vérifier que toutes les cartes ont bien été enregistrées en mémoire.

— La troisième fenêtre appelée « décodeur d'instruction » simule la production des impulsions électroniques qui activent les circuits appropriés pour l'exécution d'une instruction. Etant donné que vous vous êtes substitués aux circuits logiques, les instructions sont exprimées en français plutôt qu'en impulsions électroniques. Imaginez simplement que cette fenêtre effectue pour vous le décodage des impulsions qu'elle reçoit, afin de vous permettre de connaître l'action à accomplir.

UNITÉ DE COMMANDE



L'unité de commande d'un ordinateur suit également un cycle invariant en trois étapes :

- 1^o Elle va chercher une instruction en mémoire et l'emmagesine dans le registre d'instruction.
- 2^o Elle accroît le nombre contenu dans le compteur d'adresses, le faisant ainsi passer à la valeur donnant l'adresse de l'instruction suivante.
- 3^o Elle active le registre d'instruction pour qu'il exécute l'instruction qui vient de lui être transmise.

Pendant que le registre d'instruction exécute son cycle d'opération l'unité de commande n'intervient plus. Elle reprendra le contrôle seulement après que le registre d'instruction aura terminé une instruction.

Vous jouerez le rôle de l'unité de commande d'Ordinapoch, en suivant des yeux son cycle interne représenté par les flèches vertes. Ce faisant vous exécuterez toutes les opérations qui ont été décrites.

Mise en ordre des instructions

Dans un vrai ordinateur l'ordre de succession des impulsions produite par le registre d'instruction est très important. Dans le cas d'Ordinapoch également, une mise en séquence correcte est essentielle. Le chemin indiqué par les flèches vertes d'Ordinapoch doit être suivi scrupuleusement depuis la fenêtre du registre d'instruction jusqu'à celle du décodeur d'instruction, en passant par celle du test du contenu de l'accumulateur pour enfin revenir à la fenêtre du registre d'instruction. Cette boucle représentée par le circuit des flèches vertes, constitue ce que l'on appelle le *cyle interne* d'un ordinateur.

A la fin de chaque cycle, arrêtez-vous un instant pour bien vous rappeler que l'ordinateur accomplit le même cycle des millions de fois plus vite que vous.

Plus le cycle interne qui conditionne tout le fonctionnement d'un ordinateur est rapide, plus cet ordinateur sera capable d'accomplir des tâches inconcevables à réaliser avec des machines plus lentes. Ainsi en est-il du lancement d'une fusée lunaire. Dans ce cas, la machine doit pouvoir effectuer en « temps réel », en l'espace de quelques secondes, les millions d'opérations qui permettent de vérifier le fonctionnement de chaque pièce, de calculer la trajectoire, ou de rectifier le guidage de la fusée. Vous comprenez maintenant pourquoi la vitesse du cycle des ordinateurs modernes doit être exprimée en *microsecondes* (millionième de seconde) et même en *nanosecondes* (milliardième de seconde). Or il y a autant de nanosecondes dans une seconde que de secondes dans 30 ans !...

6 Comment écrire un programme avec Ordinapoche

Votre premier programme est très simple. Il permet d'additionner deux nombres (que vous pourrez changer après le premier essai). Nous l'avons écrit pour vous. Il ne cherche pas à vous impressionner sur la puissance d'Ordinapoche en tant qu'ordinateur, mais plutôt à vous familiariser avec son fonctionnement.

En écrivant et en utilisant ce programme, n'anticipez rien et ne sautez pas les étapes. L'omission d'une seule d'entre elles peut avoir des conséquences fâcheuses. La procédure pourra vous paraître fastidieuse mais pour les ordinateurs, elle n'est pas. Ils répètent inlassablement les mêmes opérations des milliards de fois par jour, sans se fatiguer et sans le moindre ennui ! C'est ce qui fait leur puissance.

Etant donné que ce premier programme ne comprend que 7 instruc-

tions on pourrait les écrire, les unes à la suite des autres sur une seule ligne, comme ceci : 050, 051, 150, 251, 652, 552, 900. Cependant un programme plus long écrit de cette façon, pourrait conduire à des confusions. Il est donc préférable d'écrire chaque instruction l'une au dessous de l'autre sous forme de tableau. La première

colonne indique les adresses des cases mémoires dans lesquelles chaque instruction doit être stockée. La deuxième colonne est le programme proprement dit, qui sera éventuellement chargé dans ces cases mémoires. La troisième colonne est réservée aux notes explicatives ou aux commentaires.

**Programme n° 1 :
additionner le nombre « A » au nombre « B » pour obtenir la somme « S »**

Adresse	Contenu	Commentaires
10	050	Lire « A »
11	051	Lire « B »
12	150	Mettre à 0 l'accumulateur et additionner « A »
13	251	Additionner « B » (« S » est maintenant dans l'accumulateur).
14	652	Emmagasiner « S ».
15	552	Imprimer « S ».
16	900	Arrêter et remettre à l'état initial.



Vos directives en 26 points pour vous servir d'Ordinapoche

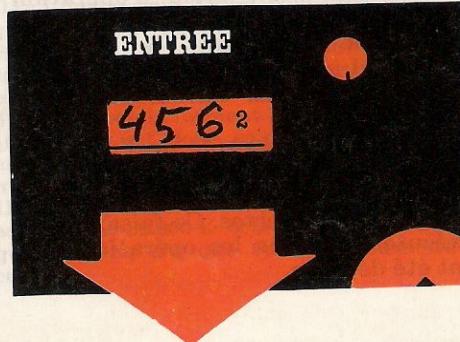
1. Ecrivez le programme de démonstration dans les cases de la MEMOIRE (vous utiliserez pour cela un stylo feutre effaçable, ou un crayon à mine tendre, pour ne pas abîmer votre Ordinapoche).

Vous commencez par écrire : dans la case n° 10 la 1^e instruction : 050 ; dans la case n° 11 la 2^e instruction : 051 ; dans la case n° 12 la 3^e instruction : 150 ; dans la case n° 13 la 4^e instruction : 251 ; dans la case n° 14 la 5^e instruction : 652 ; dans la case n° 15 la 6^e instruction : 552 ; dans la case n° 16 la 7^e instruction : 900

2. Les instructions étant prêtes, vérifiez que la ligne n° 1 du ruban-carte ENTREE apparaît bien dans la fenêtre située sous le mot ENTREE. Inscrivez-y le premier nombre à additionner : 123, par exemple.



3. Tirez sur le ruban-carte pour faire apparaître la ligne n° 2. Inscrivez-y le deuxième nombre à additionner : 456, par exemple.



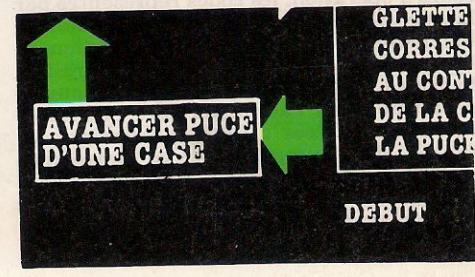
4. Remontez le ruban-carte pour que le premier nombre à additionner, « 123 » apparaisse dans la fenêtre d'ENTREE.

5. Placez une puce à la case 10 de la MEMOIRE.



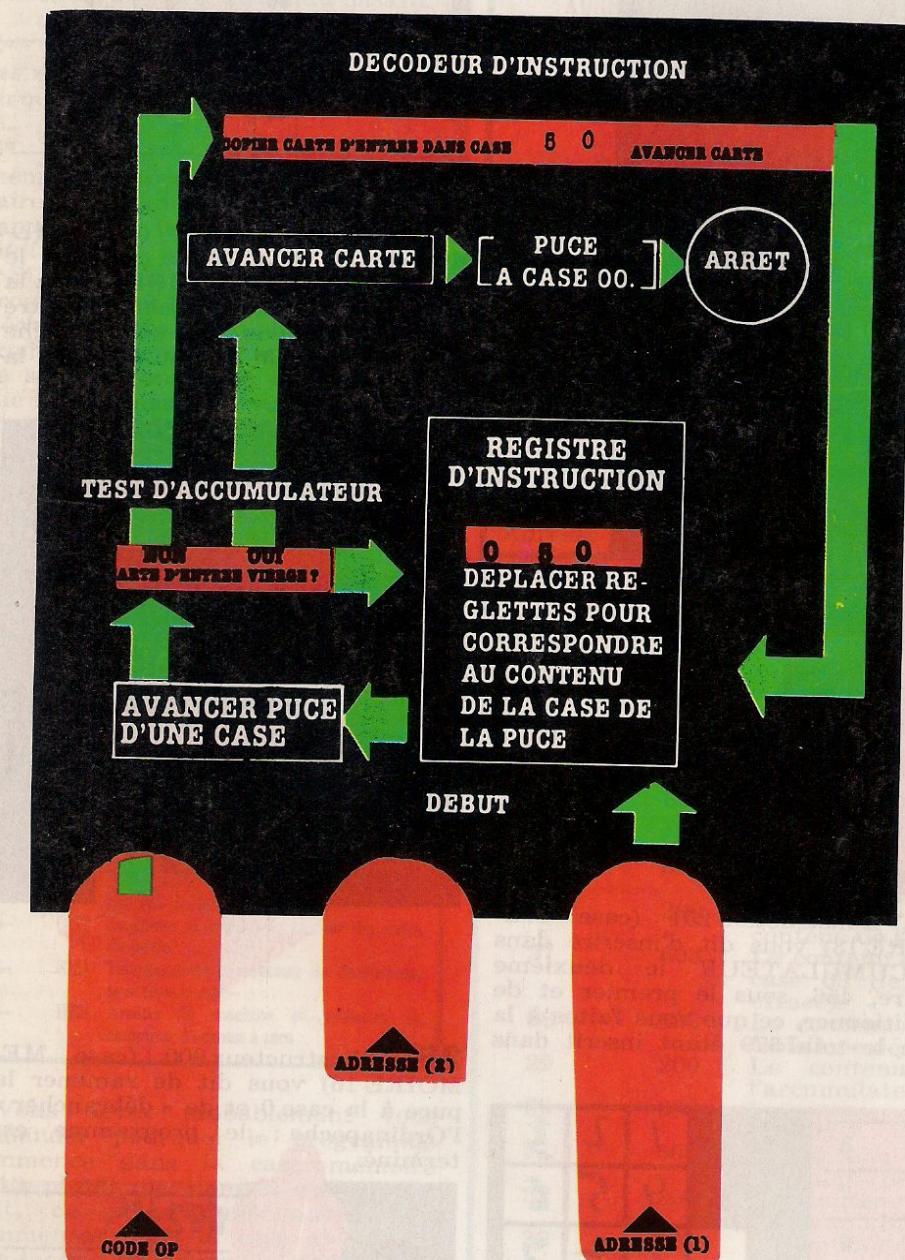
Vous allez commencer à exécuter votre programme.

6. Vous partez du mot DEBUT et vous allez suivre méthodiquement le circuit des flèches vertes.



Le programme : boucles et branchements

7. Comme il est écrit dans le rectangle REGISTRE D'INSTRUCTION, vous allez déplacer les réglettes CODE OP, ADRESSE 2, ADRESSE 1 de manière à faire apparaître dans la fenêtre le « mot programme ou instruction » inscrit dans la case Mémoire indiquée par la puce. Ce premier « mot programme » est, vous le voyez, un nombre : 050.



8. Avancez la puce d'une case (et n'oubliez pas de le faire à ce moment précis ; en général on l'oublie et cela compromet la suite des opérations !).

9. Suivez la grande flèche verte montant à gauche. Vous arrivez à la fenêtre DÉCODEUR D'INSTRUCTION qui vous indique ce que vous avez à faire. (Attention ! pour qu'il n'y ait pas d'erreur d'interprétation, les réglettes doivent être parfaitement en place).

la fenêtre d'ENTREE à la ligne 1, dans la case de Mémoire n° 51.

11. Vous inscrivez donc dans cette case le nombre 123.

29	49	69
30	50	70
31	51	71
32	52	72

12. Mais le DÉCODEUR vous demande également d'avancer le ruban-carte d'ENTREE pour faire apparaître la ligne n° 2, ce que vous faites.

13. En suivant la grande flèche verte descendant à droite vous revenez au rectangle REGISTRE D'INSTRUCTION.

Un nouveau cycle recommence

14. Vous manipulez à nouveau les réglettes comme indiqué au 7° ci-dessus. Ce qu'il faut faire apparaître dans la fenêtre est toujours indiqué par la case où se trouve la puce : elle est maintenant à la case n° 11 et le mot programme est : 051.

15. Avancez la puce d'une case.

16. Allez au DÉCODEUR D'INSTRUCTION comme indiqué au 9° ci-dessus.

17. Le programme vous dit d'inscrire dans la case Mémoire n° 51, le deuxième nombre à additionner : 456, ce que vous faites.

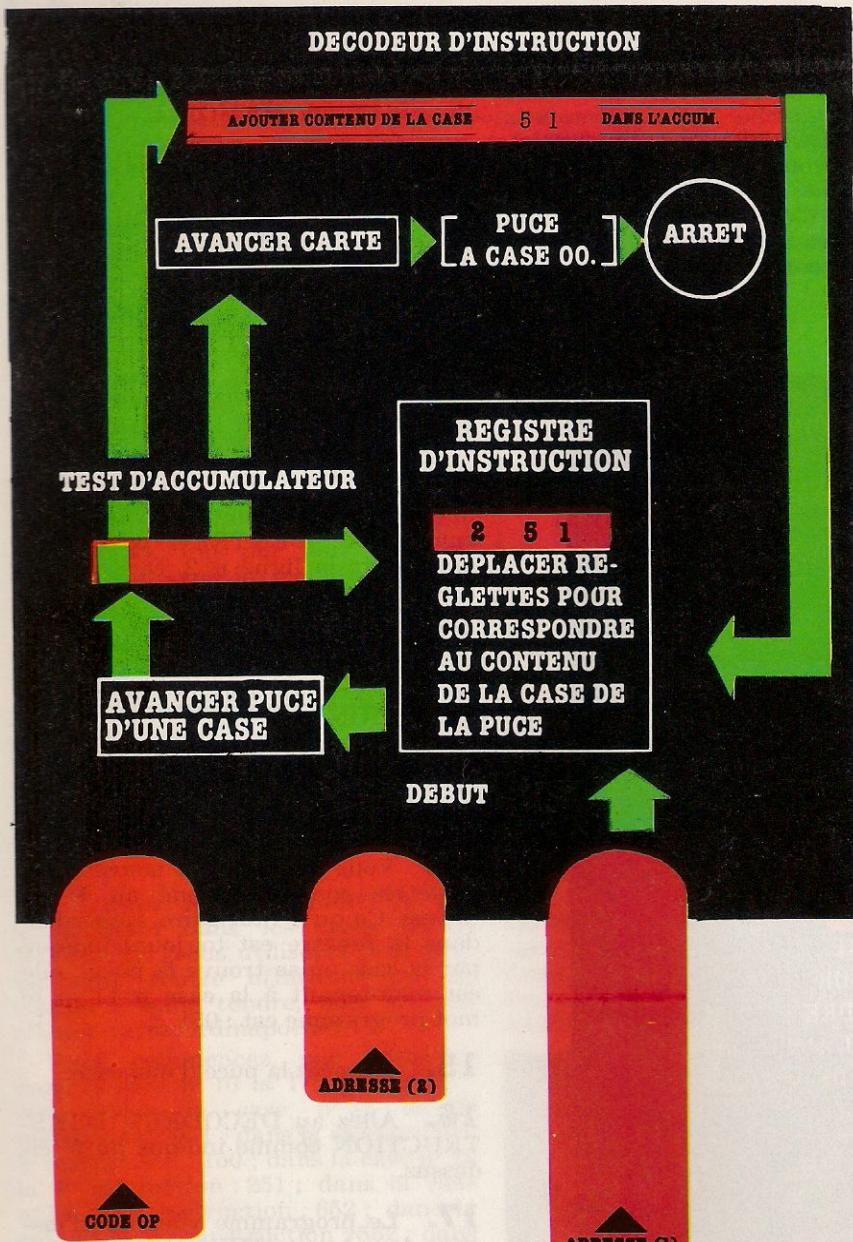
18. Il vous dit aussi d'avancer le ruban-carte d'ENTREE jusqu'à la 3° ligne (qui est vierge).

19. Retour au REGISTRE D'INSTRUCTION. Un nouveau cycle recommence. Maintenant vous avez compris le principe, il est donc possible d'abréger le détail des explications.

20. La case mémoire n° 12 vous dit de faire apparaître le mot programme : 150.

21. N'oubliez pas d'avancer la puce (on ne vous le redira plus !)

Vos directives pas à pas pour vous servir d'**Ordinapoche**



22. Comme vous le demande le DECODEUR effacez (seulement s'il y a lieu, mais il pense à tout) ce qui est dans l'ACCUMULATEUR et inscrivez en haut le premier nombre, 123, à raison d'un chiffre dans chaque petite case.

8	1	2	3
7			
6			
5			

ACCUMULATEUR

23. L'instruction 251 (case MEMOIRE 13) vous dit d'inscrire dans l'ACCUMULATEUR le deuxième chiffre, 456, sous le premier et de l'additionner, ce que vous faites à la main, le total 579 étant inscrit dans

8	1	2	3
7	4	5	6
6	5	7	9
5			

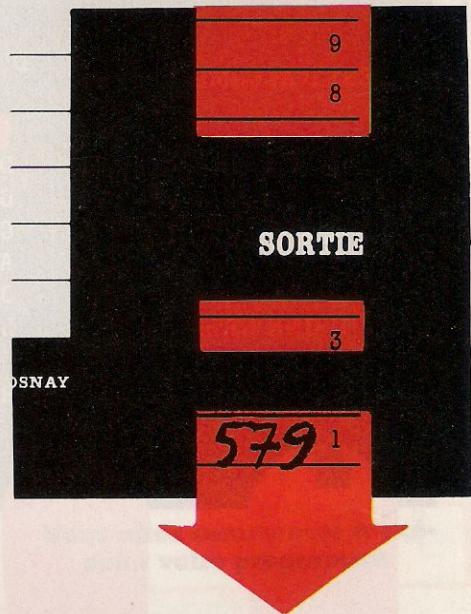
ACCUMULATEUR

les cases du bas. Ce total est appelé : « CONTENU de l'ACCUMULATEUR ».

24. L'instruction 652 (case MEMOIRE 14) vous dit de recopier le contenu de l'accumulateur soit « 579 » dans la case n° 52

53	54	55
50	50	123
51	81	456
52	52	579
53	53	73

25. L'instruction 552 (case MEMOIRE 15) vous dit de recopier le contenu de la case MEMOIRE 52 à la ligne 1 qui apparaît dans la fenêtre de SORTIE. Vous tirez sur la flèche pour faire sortir le résultat sous la fenêtre : « 579 ».



26. L'instructeur 900 (case MEMOIRE 16) vous dit de ramener la puce à la case 0 et de « débrancher » l'Ordinapoche : le programme est terminé.

652	54
552	55
900	56

7 Deux aspects importants des programmes : boucles et branchements.

COMMENTAIRE :

Ce cycle peut vous sembler long, fastidieux... et surtout incroyablement compliqué pour faire quoi ? une addition aussi simple que $123 + 456$! Mais :

1° Ce faisant vous avez pénétré la méthode de travail d'un ordinateur, une méthode qui découpe le travail en tranches sans rien laisser au hasard.

2° L'ordinateur lui effectue ce cycle des millions de fois plus vite que vous, en quelques nanosecondes peut être...

3° Une fois ce programme mis en mémoire, un ordinateur qui a appris à faire une addition est désormais capable de faire n'importe quelle addition.

Si vous avez soigneusement suivi le programme et le cycle indiqué par les flèches vertes, Ordinapoché a du produire le bon résultat sur la carte de sortie. Vous avez du remarquer que la dernière instruction, « 900 » non seulement arrête la machine, mais ramène le compteur d'adresses à zéro.

Six codes d'opérations différents ont été utilisés dans ce programme. Voici la liste de ces codes avec leurs abréviations mnémoniques et leurs explications. Les quatre codes d'opérations restants seront expliqués au fur et à mesure de leur utilisation.

Code d'opération	Abréviation	Opération
0- INP	Lire carte d'entrée dans la case-	
1- CLA	Vider l'accumulateur et y additionner le contenu de la case-	
2- ADD	Additionner le contenu de la case- au contenu de l'accumulateur.	
5- OUT	Imprimer le contenu de la case-sur carte de sortie.	
6- STO	Emmagasiner le contenu de l'accumulateur dans la case--	
9- HRS	Arrêter la machine et remettre le compteur d'adresse à zéro	

Vous devez probablement vous demander pourquoi le programme commence dans la case mémoire n° 10, plutôt que dans la case 01. En fait, ce programme aurait pu commencer dans la case n° 1. Mais une longue expérience a montré aux programmeurs qu'il était bon de laisser un certain nombre de cases vides au début d'un programme. Cela donne de l'espace lorsqu'il faut y insérer une instruction que l'on a oubliée.

LES BOUCLES

Voyons maintenant comment on peut apprendre à un ordinateur à compter jusqu'à un nombre choisi à l'avance. Regardez le programme suivant. Il va produire la série des nombres 1, 2, 3, 4, 5,... jusqu'au nombre désiré

Programme numéro 2 : comptage

ADRESSE	CONTENU
20	100
21	603
22	503
23	200
24	603
25	503
26	200
27	603
28	503
29	200
30	603
31	503
32	200
33	603
34	503

Exactement comme le programme d'addition, ce programme ne nous intéresse pas tant pour ce qu'il fait que pour la façon dont il le fait.

Adresse	Contenu	Commentaires
20	100	Le contenu de la case 00 (001) est mis dans l'accumulateur*.
21	603	Le contenu de l'accumulateur (001) est copié dans la case 03 (le contenu de l'accumulateur n'est pas effacé).
22	503	Le contenu de la case 03 est imprimé. Ceci est le premier décompte.
23	200	Le contenu de la case 00 (001) est additionné au contenu de l'accumulateur (001) éllevant la somme à 002.
24	603	Le contenu de l'accumulateur (002) est copié dans la case 03 (le contenu de l'accumulateur n'est pas effacé)**.
25	503	Le contenu de la case 03 est imprimé. Ceci est le deuxième décompte.
26	200	Le contenu de la case 00 (001) est additionné au contenu de l'accumulateur, éllevant la somme à 003.
27	603	Le contenu de l'accumulateur (003) est copié dans la case 03 (le contenu de l'accumulateur n'est pas effacé).
28	503	Le contenu de la case 03 est imprimé. Ceci est le troisième décompte.
29	200	Le contenu de la case 00 est additionné à l'accumulateur, éllevant la somme à 004.
30	603	Le contenu de l'accumulateur (004) est copié dans la case 03.
31	503	Le contenu de la case 03 est imprimé. Ceci est le quatrième décompte.
32	200	Le contenu de la case 00 est additionné à l'accumulateur éllevant la somme à 005.
33	603	Le contenu de l'accumulateur (005) est copié dans la case 03.
34	503	Le contenu de la case 03 est imprimé. Ceci est le cinquième décompte.

* Le nombre 1 (soit 001) peut être conservé en permanence dans la case 00 (inscrivez-le au crayon dans cette case mémoire).

** Remarquez que l'ancienne valeur contenue dans la case 003 (soit 001) est effacée et remplacée par la nouvelle valeur 002. On peut donc écrire dans une case de mémoire par dessus un nombre existant.

Il n'est pas nécessaire de l'exécuter sur ORDINAPOCHE. Parcourez le simplement des yeux, et vous verrez qu'il présente un sérieux inconvénient. Un coup d'œil suffit pour réaliser qu'il est d'abord beaucoup trop long : quinze instructions pour compter seulement jusqu'à 5 ! Un programme identique pour compter jusqu'à un million remplirait complètement la mémoire d'un ordinateur de très grande puissance. Un examen plus approfondi est encore plus révélateur. Après la première instruction, le programme se répète au taux de trois instructions : 603, 503, 200; 603, 503, 200; etc. Voyons ce qui se passe : Pour faire compter un ordinateur, comme vous venez de le voir, il suffit que cet ordinateur soit programmé pour additionner continuellement « un » dans l'accumulateur, puis pour mémoriser et imprimer la somme obtenue.

En réalité, compter est une fonction très importante des ordinateurs utilisée dans d'autres programmes. Le problème qui se pose est le suivant : comment l'ordinateur peut-il répéter le cycle de comptage sans qu'on ait à répéter dans le pro-

8 Pour comprendre

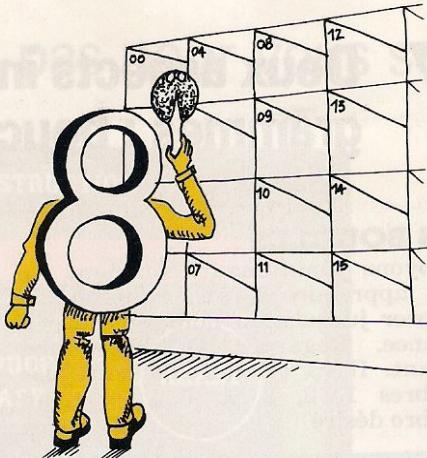
gramme la succession de chaque pas ? Une autre instruction devient nécessaire. Une instruction qui puisse être écrite une seule fois dans le programme et qui force l'ordinateur à tourner à l'intérieur d'une boucle et à revenir au début de chaque cycle d'addition-mémorisation-impression.

LE BRANCHEMENT INCONDITIONNEL

Comme tous les ordinateurs, ORDINAPOCHE possède une telle instruction. Il s'agit du code d'opération n° 8, de l'instruction saut ou plutôt de « branchement ». Cette nouvelle et importante instruction permet à la puce de revenir en arrière ou de faire un bond en avant jusqu'à la case de mémoire où l'on désire se rendre. Dans le jargon des programmeurs, cette fonction est connue sous le nom de *branchement inconditionnel*. Elle permet d'apprendre à un ordinateur à répéter autant de fois que l'on veut la même séquence d'instructions sans que l'on soit obligé d'écrire chaque étape de cette séquence des centaines ou des milliers de fois.

Comme tout ceci semble plus compliqué à décrire qu'à exécuter, utilisons tout de suite ORDINAPOCHE pour illustrer l'importance d'une telle « boucle » dans un programme de comptage.

Grâce au programme ci-dessous, un ordinateur peut compter indéfiniment. Et pourtant, il ne contient que



cinq instructions ! Exécutez-le sur ORDINAPOCHE jusqu'au nombre 3 ou 4 pour vous assurer qu'il fonctionne. Placez la puce en position de départ, par exemple à la case 21.

Le code d'opération 8 présente également un autre avantage (qui n'a pas été utilisé dans le programme ci-dessus) : non seulement il permet au compteur d'adresse de commencer une boucle en se branchant en dehors de la séquence habituelle, mais il enregistre également la valeur du compteur de façon que celui-ci puisse retourner où il se trouvait au moment où l'ordinateur a commencé à parcourir la boucle. Pour cela, le code d'opération 8 introduit une sous-instruction qui permet d'enregistrer la dernière adresse de la puce (avant le branchement) dans la case 99. Cette méthode sera détaillée un peu

Un ordinateur enfermé dans une boucle se trouve dans une situation grave. Et comme tous les programmes sont à la merci de ce genre d'erreur, il faut trouver un moyen d'éviter qu'elle ne conduise à une consommation trop élevée de « temps machine ». Pour cela, le temps moyen de passage des programmes en machine est évalué, ce qui permet d'« éjecter » un programme qui dépasse de manière flagrante le temps qui lui est alloué.

Pour sortir d'une boucle, l'ordinateur doit être capable de « prendre une décision » fondée sur un critère prédéterminé. Ce critère peut être spécifié par le programmeur mais la capacité à prendre cette décision est incorporée dans le « matériel » de l'ordinateur.

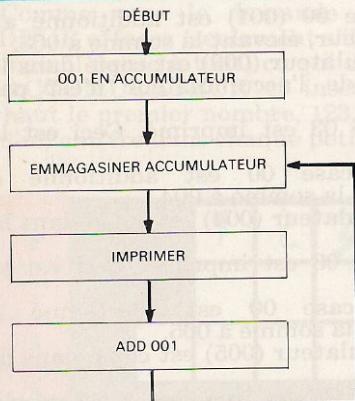
Etant donné qu'un ordinateur pris dans une boucle répète indéfiniment les mêmes instructions, la « décision » qu'il prend doit être fondée sur une nouvelle instruction qui *interrompra le cycle*. L'ordinateur doit savoir exactement à quel moment faire usage de cette nouvelle instruction, détecter un certain changement prédéterminé se produisant en son sein au cours de son fonctionnement et réagir immédiatement. Ce changement peut être (comme c'est très souvent le cas) le changement de signe d'un nombre contenu dans l'accumulateur.

Prenons par exemple un ordinateur programmé pour compter à rebours à partir de 100. Pour cela, le programme place 100 dans l'accumulateur et soustrait de façon répétitive le nombre 1 au moyen d'une boucle. Au bout de 100 soustractions répétées, l'accumulateur *passera de zéro à moins un*. Zéro est arbitrairement défini comme un nombre positif. Donc, si le comptage doit être de 100 exactement, il doit commencer à 99. A ce moment précis, le changement de signe est détecté par le circuit approprié et permet de passer à une nouvelle instruction. C'est exactement ce qui se produit dans Ordinapoch : la nouvelle instruction est le code d'opération n° 3. Son abréviation en code mnémonique est « TAC » (de l'anglais : Test Accumulator Content ou Tester le contenu de l'accumulateur). Le programme ci-après permet d'illustrer l'utilisation de cette instruction.

Ce programme permet de déclencher le départ d'une fusée après un très court « compte à rebours » (de 4 à 0). Le nombre « moins quatre » (-4) est placé dans l'accumulateur et le programme ajoute « un » jusqu'à ce

Programme n° 3

ADRESSE	CONTENU	COMMENTAIRES
21	100	Mise à zéro et addition du contenu de la case 00.
22	603	Emmagasiner le contenu de l'acc. dans la case 03.
23	503	Imprimer le contenu de la case 03.
24	200	Additionner à l'accumulateur le contenu de la case 00.
25	822	Sauter à l'instruction de la case 22.



Ordinogramme du programme de calcul utilisant une boucle.

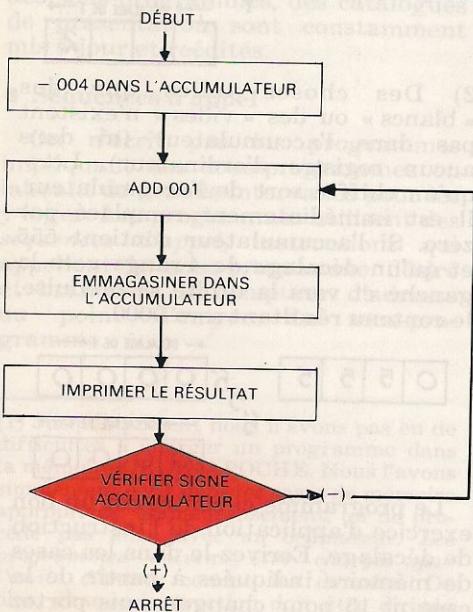
la puissance et la souplesse de l'ordinateur.

que l'accumulateur arrive à zéro. Puisque zéro est (pour Ordinapoché) un nombre positif, le signe de l'accumulateur change en arrivant à zéro et la boucle d'addition est brusquement interrompue. Vous pouvez utiliser un programme similaire à celui de l'addition, mais en introduisant simplement une instruction supplémentaire et en changeant les données de départ.

Exécutez ce programme sur Ordinapoché en observant attentivement ce qui se passe lorsque vous changez le signe de l'accumulateur. Notez également la façon dont le code d'opération n° 3 établit une boucle et comment il l'interrompt lors du changement de signe de l'accumulateur. Après chargement du programme en mémoire, faites partir la puce à la cellule 10.

Programme n° 4 Compte à rebours pour le lancement d'une fusée (1)

Adresse	Contenu	Commentaires
00	+ 001	Données
01	- 004	Données
10	101	Mettre - 004 dans l'accumulateur
11	200	Additionner 001
12	650	Emmagasiner le contenu de l'accumulateur dans la case 50.
13	550	Imprimer le contenu de la case 50.
14	311	Tester l'accumulateur : - si négatif, aller à la case 11, - si positif, aller à la case 15.
15	900	Arrêter et remettre à l'état initial.



(1) Le lancement se produit lorsque la sortie de l'ordinateur donne 000.

Comparaison des instructions de branchement conditionnel et inconditionnel

Comme le démontrent les deux programmes précédents, le branchement conditionnel (code d'opération 3) et le branchement inconditionnel (code d'opération 8) modifient tous deux la valeur du compteur d'adresse, c'est-à-dire qu'ils branquent la puce à une adresse située hors de la séquence normale. La principale différence entre ces deux instructions réside dans le fait que le branchement inconditionnel provoque toujours un branchement alors que le branchement conditionnel ne provoque ce branchement que lorsque le signe de l'accumulateur est négatif. Le branchement conditionnel offre non seulement un moyen de sortir d'une boucle mais aussi de choisir des séquences d'instruction en fonction des résultats obtenus auparavant. Cette possibilité de choix est connue sous le nom de *point de branchement*.

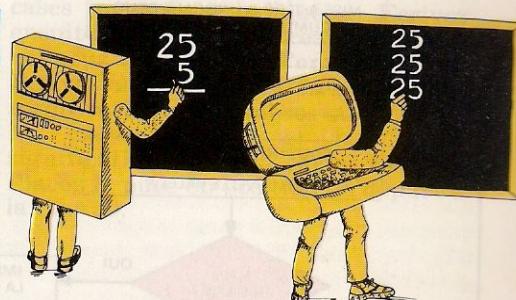
LA MULTIPLICATION

Il y a deux façons de multiplier avec l'ordinateur — l'une économique, l'autre dispendieuse.

Bien sûr, les choses ne sont pas si simples, mais il est vrai que le coût d'un ordinateur est assez bon indice de sa façon de multiplier. En règle générale, les gros ordinateurs, plus dispendieux, utilisent des circuits logiques qui permettent de multiplier (ou de diviser) directement — un peu comme nous le faisons.

Les petits ordinateurs ne possédant pas de tels circuits, utilisent la méthode de l'addition successive. Ils placent le plus grand des deux nombres à multiplier dans l'accumulateur et additionnent ce nombre à lui-même « N » fois, « N » étant égal au plus petit nombre, moins un.

Par exemple, pour multiplier 25 par 5, ils placent 25 dans l'accumulateur et, au moyen d'une boucle, lui additionnent quatre fois le nombre.



Comme l'ordinapoché a été conçu selon les modèles les moins chers, lui aussi multiplie par additions successives. Cette méthode est illustrée dans le programme numéro 5 en page 38.

Remarquez que ce programme utilise le code d'opération 8 pour compléter la boucle d'addition et le code d'opération 3 pour en sortir, que « N » est vérifié durant chaque cycle et que la boucle n'est pas brisée tant que « N » n'est pas négatif. On désigne cette méthode sous le nom de boucle indexée et l'index est égal à « N ». La même méthode peut s'employer dans n'importe quel programme exigeant « N » répétitions d'une procédure particulière. Etant donné que « N » est introduit comme donnée plutôt que comme partie intégrante du programme, il n'est pas nécessaire de corriger de tels programmes lors d'un changement de « N ». (1)

Dans les ordinogrammes, ils sont représentés par une figure en forme de losange et indiquent toujours le point où une décision doit être prise. Pour voir comment elles s'utilisent, reportez-vous à la page 25 et jetez à nouveau un coup d'œil sur l'ordinogramme du « changement de pneu ».

Quand vous aurez plus d'expérience, vous pourrez écrire cet ordinogramme en langage machine d'Ordinapoché. En attendant, essayez de trouver une façon de convertir les « oui » et les « non » de l'ordinogramme en signes « plus » ou « moins » d'un accumulateur d'ordinateur. (1).

(1) Pour ceux qui s'intéressent dès maintenant aux langages de programmation (ou qui en connaissent les bases) l'instruction de branchement inconditionnel en BASIC est le GOTO (Aller à) et l'instruction de branchement conditionnel IF, THEN (Si, alors).

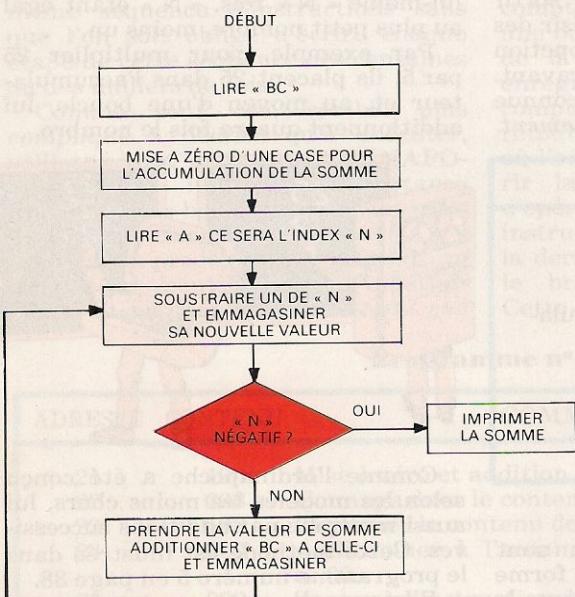
(1) en BASIC, la boucle tournera 10 fois grâce à l'instruction : FOR N = 1 TO 10 NEXT N.

Programme n° 5 : multiplication par un nombre d'un seul chiffre

LE DÉCALAGE DE CHIFFRES

Adresse	Contenu	(multiplicateur) A x BC (multiplicande)	Commentaires
07	068	Lire « BC » et placer dans case 68.	
08	404	Mise à zéro* de l'acc.	Mise à zéro de la case 69 pour emmagasinage futur de la somme.
09	669	Emmagasiner acc. (zéro) dans case 69.	
10	070	Lire « A » et le placer dans la case 70.	Ceci représente « N ».
11	170	Placer « N » dans acc.	
12	700	Soustraire 1 de « N ».	
13	670	Emmagasiner « N » modifié.	
14	319	Vérifier signe de l'acc.	
15	169	Mise à zéro de l'acc. et placer contenu de la case 69 (ancienne somme).	
16	268	Additionner « BC » à l'acc.	
17	669	Emmagasiner la somme modifiée dans case 69.	
18	811	Aller à la case 11.	
19	569	Imprimer (le produit de « A » x « BC »).	
20	900	Arrêter et remettre à l'état initial.	

* Voir en colonne de droite la manière dont cette instruction fait la mise à zéro de l'accumulateur.



Ordinogramme d'une multiplication à un seul chiffre.

Programme n° 6 : inversion de l'ordre d'un nombre "ABC"

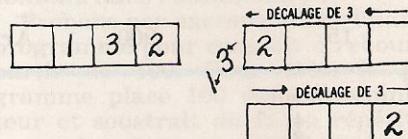
Code d'opération 4 Instruction de décalage

Comme le programme suivant le démontre, ORDINAPOCHE peut non seulement inverser des nombres mais aussi les manipuler de plusieurs façons. Pour cela, il utilise le code d'opération 4 — l'instruction de décalage.

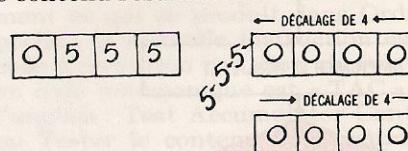
Son rôle, en bref, est de décaler un nombre contenu dans l'accumulateur d'un nombre « X » de rangs vers la gauche puis d'un nombre « Y » de rangs vers la droite. La valeur de « X » et « Y » est spécifiée par le deuxième et le troisième chiffre de l'instruction de décalage. C'est la seule instruction dont les deux derniers chiffres ne correspondent pas à une adresse mémoire.

Avant de pouvoir l'utiliser convenablement, vous devez comprendre deux choses :

1) Les chiffres dépassant la capacité de l'accumulateur sont irrécupérables. Prenons par exemple le nombre 132 comme contenu de l'accumulateur et la valeur 433 au registre d'instructions. Celui-ci demande que le nombre se trouvant dans l'accumulateur soit décalé de trois rangs vers la gauche puis de trois rangs vers la droite. Le contenu final sera-t-il identique au contenu initial ? Pas du tout ! Lorsque le décalage vers la gauche pousse les chiffres 1 et 3 hors de l'accumulateur ils sont perdus à jamais. Lors du décalage à droite, seul le chiffre 2 revient à sa valeur initiale.



2) Des choses telles que des « blanches » ou des « vides » n'existent pas dans l'accumulateur (ni dans aucun registre d'ordinateur). Lorsqu'un chiffre sort de l'accumulateur, il est immédiatement remplacé par zéro. Si l'accumulateur contient 555 et qu'un décalage de 4 rangs vers la gauche et vers la droite se produise, le contenu résultant sera 0000.



Le programme ci-contre est un bon exercice d'application de l'instruction de décalage. Ecrivez-le dans les cases de mémoire indiquées à partir de la case n° 15 pour changer, puis portez le nombre de trois chiffres à inverser sur la carte d'entrée n° 1. Placez la puce à la position de départ 15.

ADRESSE	CONTENU	COMMENTAIRES
15	039	Lire « abc » et le placer dans case 39.
16	139	Mise à zéro de l'accumulateur et y placer « abc ».
17	431	Décaler l'accumulateur pour avoir « c00 ».
18	640	Emmagasiner l'accumulateur dans case 40.
19	139	Mise à zéro de l'accumulateur et add. « abc » dans l'accumulateur.
20	413	Décaler l'accumulateur pour produire « 00a »
21	240	Additionner le contenu de la case 40 pour avoir « c0a » dans l'accumulateur.
22	640	Emmagasiner l'accumulateur dans la case 40.
23	139	Mise à zéro et additionner « abc ».
24	423	Décaler l'accumulateur pour avoir « 00b ».
25	410	Décaler l'accumulateur pour avoir « 0b0 ».
26	240	Additionner la case 40 pour avoir « cba » en accumulateur.
27	640	Emmagasiner l'accumulateur dans la case 40.
28	540	Imprimer le contenu de la case 40.
29	900	Arrêter et remettre à l'état initial.

LES SOUS-PROGRAMMES

A l'époque des premiers ordinateurs, les programmeurs ne disposaient pas de bibliothèques de programmes enregistrés auxquelles se reporter. A chaque fois qu'ils s'attelaient à un nouveau programme, ils devaient partir de zéro.

Si une partie du programme impliquait des sinus, cosinus, racines carrées ou l'une des mille autres routines mathématiques, il leur fallait écrire chaque étape de cette routine.

Au bout d'un certain temps, ils réalisèrent qu'ils recréaient sans cesse les mêmes routines. C'était un immense gaspillage de temps et d'énergie créatrice; et c'est à partir de ce jour que naquit le concept du sous-programme.

• Qu'est-ce qu'un sous-programme ?

Un sous-programme est simplement une partie d'un programme mémorisée en général sur bande magnétique de façon à pouvoir être facilement utilisée par n'importe quel programme.(1)

Il existe des sous-programmes couvrant depuis des fonctions aussi triviales que les sinus et cosinus jusqu'à des fonctions très ésotériques. Leur entité réside dans le fait que, une fois écrits et enregistrés, ils n'ont plus besoin d'être écrits à nouveau.

Quelquefois, un programme complet s'avère utile dans la solution partielle d'un problème plus grand. Dans ce cas, on peut l'employer comme sous-programme du programme principal.

En raison de la prolifération rapide des sous-programmes, des catalogues de présentation sont constamment mis à jour et réédités.

• Séquences d'appel

On n'écrit pas les programmes impliquant des sous-programmes en laissant de grands intervalles pour les y loger. Les programmes utilisent des séquences d'appel, instructions qui appellent le sous-programme désiré et branchent le compteur d'adresse au point d'entrée du sous-programme.

La séquence d'appel offre également la possibilité de transférer des données du programme au sous-programme et vice-versa. Elle doit aussi noter l'adresse à laquelle le programme principal devra retourner après déroulement du sous-programme. Pour cela, Ordinapoch utilise le code d'opération 8. Dans le programme qui suit, vous allez appliquer cette instruction dans sa totalité pour la première fois, y compris la partie qui vous demande d'écrire le numéro de case de la puce dans la case 99.

• Double précision

La précision de toute valeur numérique dépend du nombre de chiffres significatifs utilisés. Si vous annoncez qu'un objet a une longueur de 3,62958 cm, vous êtes beaucoup plus précis que quelqu'un qui se limite à donner une longueur de 3,63 cm.

Ainsi, la précision des ordinateurs semblerait être limitée par le nombre de chiffres qu'un mot-mémoire peut contenir.

Heureusement, ceci n'est pas toujours vrai. Assez curieusement, les petits ordinateurs peuvent être programmés pour simuler la capacité plus élevée de leurs grands frères. Mais ils le font aux dépens de la vitesse car ils doivent exécuter beaucoup plus d'opérations pour accomplir la même chose. Naturellement, il existe des limites pratiques au delà desquelles la simulation ne peut être poussée plus loin. En termes de coût par unité d'opération, il est encore moins onéreux de résoudre les gros problèmes sur des ordinateurs de grande puissance.

• Sous-programme en double précision pour ORDINAPOCHE

Ce programme d'addition en double précision illustre tous les points considérés ci-dessus. Il vous permet de traiter des nombres de 6 chiffres sur le « matériel » d'Ordinapoch conçu pour 3 chiffres. Lorsque vous aurez terminé, comparez le temps passé à additionner deux nombres de

6 chiffres avec le temps passé à faire l'addition de deux nombres à 3 chiffres. Vous appréciez alors le véritable sens de « aux dépens de la vitesse ».

A la base, la méthode utilisée pour manipuler les nombres de 6 chiffres est d'emmager les trois chiffres les plus significatifs dans une case de mémoire et les trois chiffres les moins significatifs dans une autre. Les deux cases seront adjacentes, les chiffres les plus significatifs allant dans une case de numéro impair et les moins significatifs dans la case de numéro pair suivante. On peut par exemple emmagasiner 163 742 en mettant 163 dans la case 21 et 742 dans la case 22. Les détails de toute cette comptabilité seront traités par le sous-programme.

Par ailleurs, nous pourrions écrire, des sous-programmes de 6 chiffres pour chacune des dix instructions d'Ordinapoch. Dans ce cas, tout programme en simple précision pourrait être converti en double précision par substitution des programmes en double précision à chaque instruction courante.

Emmagasiner le sous-programme et le programme principal dans les cases mémoires indiquées. Ecrivez ensuite les deux nombres à additionner sur les cartes d'entrée 1 à 4. Ecrivez les trois chiffres les plus significatifs des nombres A et B sur les cartes 1 et 3 et les moins significatifs sur les cartes 2 et 4. Placez la puce en position de départ à la case 50.

Programme principal

ADRESSE	CONTENU	COMMENTAIRES
50	095	
51	096	
52	097	Entrée et séquence d'appel
53	098	
54	886	
55	659	
56	559	Sortie
57	598	
58	900	Arrêter et remettre à l'état initial,

Programme n° 7 : sous-programme pour "A" + "B" = Somme

ADRESSE	CONTENU	COMMENTAIRES
86	199	Préparer la sortie.
87	694	
88	196	
89	298	Additionner les chiffres les moins significatifs.
90	698	
91	403	Décaler le dépassement de capacité vers la droite et additionner les chiffres les plus significatifs.
92	295	
93	297	
94	8--	Retourner au programme principal (-- sera l'adresse de la dernière instruction plus un).

(1) Jusqu'à présent nous n'avons pas eu de difficultés à charger un programme dans la mémoire ORDINAPOCHE. Nous l'avons simplement écrit dans les cases de mémoire appropriées. Malheureusement, on ne procède pas ainsi avec un ordinateur. Les programmes doivent être chargés par l'unité d'entrée comme les données. De plus, chaque instruction doit être dirigée vers la bonne adresse. Cela nécessite un programme spécial de chargement. Pour éviter d'alourdir le texte nous ne traiterons pas ce sujet

DES PROGRAMMES POUR Écrire DES PROGRAMMES

• ASSEMBLEURS

Il suffit de jeter un coup d'œil sur la liste des codes opérations d'Ordina-poche pour voir qu'ils sont conçus pour l'arithmétique. Malgré cela, nous avons essayé de le programmer pour résoudre un problème non arithmétique comme jouer au jeu de Nim (voir Annexe).

Toutefois, écrire de tels programmes pour Ordinapoche — sans parler de leur exécution — serait extrêmement fastidieux. Comme le montre notre programme du jeu de Nim perfectionné, l'un des principaux problèmes est de savoir où se trouvent en mémoire les différents points d'information. Nous devons nous rappeler non seulement de ce que N, P, C et R représentent mais aussi de l'endroit où ils se trouvent rangés. Ce genre de comptabilité interne peut être très difficile à tenir lorsqu'on a affaire à des programmes complexes.

Dans notre dernier programme, détaillé en annexe, la mise à jour de N exige trois instructions qui entraînent (1) l'envoi de N dans l'accumulateur, (2) la soustraction de P de N et (3) la mise en mémoire du résultat. Pour écrire ces instructions, il faut se souvenir des adresses qui ont été assignées à N et P ainsi que des codes d'opérations. Une fois que tout cela est clair et précis, nous écrivons les instructions 114, 715 et 614.

Maintenant, si les ordinateurs pouvaient effectuer toute cette comptabilité pour nous, la programmation serait beaucoup plus simple. A la place de 114, par exemple, nous pourrions nous contenter d'écrire CLA N et laisser à l'ordinateur le soin de trouver la position de N et le code d'opération de CLA.

Heureusement, il est possible de programmer les ordinateurs pour ces tâches précises. Les programmes utilisés dans ce cas s'appellent des *assembleurs*. En réalité, les assembleurs peuvent accomplir beaucoup plus de choses que nous ne pouvons en évoquer ici. Ils s'occupent, entre autres, des sous-programmes et des séquences d'appel. L'un dans l'autre, ils représentent une aide précieuse pour les programmeurs — à tel point que, très souvent, ils résident en permanence dans la mémoire de l'ordinateur.

• COMPILATEURS

Les assembleurs sont des programmes conçus pour écrire des progra-

mes. Ils prennent un programme écrit en langage assembleur et le transcrivent dans l'alphabet plus élémentaire du langage machine. Ainsi, le langage assembleur est plus évolué que le langage machine, en ce sens qu'il est plus facilement compréhensible par l'homme. Par bonheur, le processus d'écriture d'un programme capable de traduire un langage de niveau supérieur dans un langage moins évolué peut être mené plus loin par les *compilateurs*.

Les compilateurs sont utilisés pour transcrire en langage assembleur des programmes d'un niveau encore plus élevé que ceux que traitent les assembleurs. Ils permettent de simplifier encore davantage le travail fastidieux de la programmation. Tandis que les programmes d'assemblage nécessitent une instruction pour chaque instruction en langage machine, les compilateurs peuvent traduire une instruction écrite dans un langage d'un haut niveau (FORTRAN, BASIC, COBOL) en plusieurs instructions en langage assembleur. Ainsi, le nombre d'instructions que doit écrire un programmeur se trouve considérablement réduit.

L'un des compilateurs les plus connus et les plus largement utilisés est le FORTRAN -sigle de FORMula TRANslator. Il est conçu pour des applications mathématiques et son langage ressemble à celui de l'algèbre.

Avec FORTRAN, un programmeur désirant résoudre une équation telle que $J = K + M - N + 2$ écrirait son équation $J = K + M - N + 2$. Le compilateur de FORTRAN traduirait alors cette instruction en une instruction en langage machine. Pour Ordinapoche, ces instructions seraient :

CLA K ADD M SUB N
ADD 2 STO J

L'assembleur les traduirait ensuite en langage machine et leur assignerait des adresses en mémoire en procédant de la façon suivante (1) :

20	151	23	270
21	252	24	650
22	753		

A ce point, l'ordinateur commandera la production d'un jeu de cartes perforées ou l'inscription sur une bande magnétique, contenant le programme complet. Si cela était nécessaire, il imprimerait simultanément *le programme dans les trois langages* de façon que le programmeur puisse effectuer un contrôle. La sortie sur imprimante ou sur l'écran de visualisation comprendrait aussi une table des affectations en mémoire temporaire et une table des littéraux qui se présenterait comme suit :

Table des affectations en mémoire temporaire
50 J 52 M
51 K 53 N

Table des littéraux 70 002

Si des erreurs étaient commises, les diagnostics pourraient également figurer sur l'imprimante ou apparaître sur l'écran de visualisation. Ce sont des observations sur des erreurs banales de programmation comme l'omission de parenthèses ou l'utilisation de symboles non définis.

Maintenant que le programmeur sait où sont placés les programmes et les données et (le cas échéant) où se trouvent les erreurs, il peut entreprendre la mise au point de son programme. Dès qu'il localise ses erreurs, il retire simplement les cartes erronées de son paquet de cartes, en perfore de nouvelles et introduit le programme édité dans un lecteur de cartes. De même en tapant sur les touches du clavier, il peut effectuer des corrections dans la mémoire ou sur la bande magnétique grâce à un programme spécial d'EDITION. Il peut répéter ce processus deux ou trois fois avant que toutes les imperfections disparaissent, mais, une fois que tout est correct, l'ordinateur exécutera son programme avec la vitesse et la précision qui font de cette époque, l'Ere de l'Ordinateur.

(1) Un autre compilateur très utilisé est celui du langage BASIC : si l'utilisateur souhaite additionner A et B, il écrira en BASIC un court programme dont les traductions assembleur et langage machine d'Ordinapage figurent dans le tableau ci-dessous.

UTILISATEUR	BASIC	ORDINAPOCHE (assembleur)	ORDINAPOCHE (langage machine)
$A + B = S$	INPUT A INPUT B $A + B = S$ PRINT S END	INP A INP B CLA A ADD B STO S OUT S HRS	050 051 151 251 652 552 900

ANNEXE

Pour apprendre à concevoir un programme, LE JEU DE NIM A UNE RANGÉE

Les trois principales étapes de la préparation d'un programme sont :

- 1° définir clairement le problème ;
- 2° trouver l'algorithme approprié ;
- 3° exprimer cet algorithme dans un langage compris par l'ordinateur.

Pourquoi le jeu de Nim à une rangée ?

Nous avons choisi ce jeu parce que, mieux que la plupart des jeux, il peut être défini simplement. Ses règles, ses stratégies et ses déplacements peuvent être bien précisés. De plus, c'est un problème qui est suffisamment simple pour qu'on puisse le comprendre globalement assez facilement. Enfin, les nombres s'y présentent d'une façon naturelle et cela nous aidera à établir un programme simple pour Ordinapoche.

Nous allons analyser le jeu et écrire une série de programmes qui nous permettront de nous mesurer à Ordinapoche. Chacun des nouveaux programmes permettra à Ordinapoche d'accomplir une tâche différente exigée par le jeu. Certaines de ces tâches ne sont pas visées par les dix instructions d'Ordinapoche. Cela veut dire que nous aurons à faire preuve d'ingéniosité en concevant les programmes.

Règles du jeu de Nim à une rangée

Dix cailloux forment une seule rangée entre les deux joueurs. Quand le tour d'un joueur arrive, *il peut enlever un, deux ou trois cailloux*, pourvu que son adversaire n'ait pas enlevé le même nombre de cailloux à son tour.



En d'autres termes, si votre adversaire (Ordinapoche) vient d'enlever deux cailloux, vous pouvez en enlever un ou trois, mais pas deux. Un joueur perd lorsque :

- 1° il ne lui reste plus de cailloux à enlever ;
- 2° il ne lui reste plus qu'un caillou et son adversaire vient d'enlever un caillou.

La stratégie est donc simple

Tout coup peut être défini par un nombre de deux chiffres tel que $3\blacktriangleright 4$. Le premier chiffre représente le nombre de cailloux retiré par le joueur, et le deuxième chiffre représente le nombre de cailloux restant dans la rangée. Par exemple, si le premier joueur enlève un caillou (de la rangée initiale de dix), le coup est défini par $1\blacktriangleright 9$. Si le deuxième joueur prend maintenant deux cailloux (en laissant sept), son jeu est défini par $2\blacktriangleright 7$.

Le jeu se termine avec un de ces quatre coups gagnants : $1\blacktriangleright 0$; $2\blacktriangleright 0$; $3\blacktriangleright 0$; ou $1\blacktriangleright 1$.

Analysons le problème plus en détail

Il est évident qu'un coup doit être soit un coup gagnant, soit un coup perdant. Si un coup ne laisse à un des joueurs qu'un choix de coup perdant, le résultat sera assez évident. Mais si un coup donne à l'un des joueurs le choix de coups gagnants ou perdants, nous admettons qu'il est assez malin pour choisir toujours le coup gagnant. Nous pouvons donc, à partir des quatre dernières combinaisons gagnantes, revenir en arrière pour identifier tous les coups possibles pour un gagnant ou un perdant.

Ainsi, comme $1\blacktriangleright 0$ est un coup gagnant parce qu'il ne laisse à l'adversaire aucune possibilité de jeu, $2\blacktriangleright 1$ et $3\blacktriangleright 1$ sont des coups perdants parce qu'ils permettent au prochain joueur de faire le coup gagnant $1\blacktriangleright 0$.

De même, nous pouvons, à partir du coup gagnant $2\blacktriangleright 0$, revenir en arrière pour trouver les coups perdants $1\blacktriangleright 2$ et $3\blacktriangleright 2$. Nous pouvons aussi partir du coup gagnant $3\blacktriangleright 0$ pour arriver aux coups perdants $1\blacktriangleright 3$ et $2\blacktriangleright 3$. Et, finalement, nous pouvons, à partir du coup gagnant $1\blacktriangleright 1$, trouver les coups perdants $2\blacktriangleright 2$ et $3\blacktriangleright 2$.

Comme il sera nécessaire de représenter tous les coups possibles, il sera préférable de les mettre sous forme de tableau.

COUPS GAGNANTS			COUPS PERDANTS		
$1\blacktriangleright 0$	$2\blacktriangleright 0$	$3\blacktriangleright 0$	$1\blacktriangleright 2$	$2\blacktriangleright 1$	$3\blacktriangleright 1$
$1\blacktriangleright 1$			$1\blacktriangleright 3$	$2\blacktriangleright 2$	$3\blacktriangleright 2$

Toujours en revenant en arrière (partant cette fois des coups perdants pour arriver aux gagnants), nous pouvons trouver cinq autres coups gagnants : $3\blacktriangleright 3$; $1\blacktriangleright 4$; $2\blacktriangleright 4$; $3\blacktriangleright 4$; et $1\blacktriangleright 5$. Nous pouvons aussi, à partir de ces cinq coups, trouver sept autres coups perdants : $2\blacktriangleright 5$; $3\blacktriangleright 5$; $1\blacktriangleright 6$; $2\blacktriangleright 6$; $3\blacktriangleright 6$; $1\blacktriangleright 2$; et $2\blacktriangleright 7$. Et, enfin, en partant de ces coups perdants, nous trouvons cinq autres coups gagnants qui permettent de compléter notre tableau :

COUPS GAGNANTS			COUPS PERDANTS		
$1\blacktriangleright 0$	$2\blacktriangleright 0$	$3\blacktriangleright 0$			
$1\blacktriangleright 1$			$2\blacktriangleright 1$	$3\blacktriangleright 1$	
			$1\blacktriangleright 2$	$2\blacktriangleright 2$	$3\blacktriangleright 2$
			$1\blacktriangleright 3$	$2\blacktriangleright 3$	
$1\blacktriangleright 4$	$2\blacktriangleright 4$	$3\blacktriangleright 4$			
$1\blacktriangleright 5$			$2\blacktriangleright 5$	$3\blacktriangleright 5$	
			$1\blacktriangleright 6$	$2\blacktriangleright 6$	$3\blacktriangleright 6$
			$1\blacktriangleright 7$	$2\blacktriangleright 7$	
$1\blacktriangleright 8$	$2\blacktriangleright 8$	$3\blacktriangleright 7$			
$1\blacktriangleright 9$					

Un examen plus approfondi de notre tableau montre que le joueur qui débute devrait gagner, vu que les trois premiers coups possibles ($1 \blacktriangleright 9$, $2 \blacktriangleright 8$, $3 \blacktriangleright 7$) sont dans les colonnes des coups gagnants.

Maintenant, que nous connaissons la valeur de chaque coup, nous pouvons emmagasiner cette information dans la mémoire d'Ordinapoche et concevoir un programme simple qui permette de la consulter au besoin. C'est ce qu'on entend par *programme de consultation de table*.

Ordinapoche, comme nous le savons, utilise des instructions à trois chiffres. Le premier chiffre sert à identifier le joueur. Zéro représente Ordinapoche, et 5 l'adversaire humain d'Ordinapoche.

Le deuxième chiffre indique le nombre de cailloux enlevés, et le troisième le nombre de cailloux restant dans la rangée.

Voici, en exemple, cinq coups d'un jeu typique :

- 028 Ordinapoche prend deux cailloux et en laisse huit.
- 517 Le joueur prend un caillou, en laisse sept.
- 034 Ordinapoche en prend trois et en laisse quatre.
- 513 Le joueur en prend un, en laisse trois.
- 030 Ordinapoche en prend trois, n'en laisse pas et gagne.

L'utilisation de ces conventions et du programme de consultation de table nous permet d'élaborer ce qui est peut-être le plus court programme de jeu qui ait jamais été rédigé. Le voici dans sa totalité.

Programme numéro 8 : jeu de Nim (dix cailloux)

ADRESSE CONTENU	COMMENTAIRES
00 001	Lire une carte d'entrée (cette instruction est déjà dans la machine).
01 529	Ceci est utilisé seulement lorsque Ordinapoche joue en premier. L'instruction 529 doit être emmagasinée en case 01 au début de chaque jeu.
02 900	Arrêt et mise à zéro.

La table de consultation suivante doit être placée dans la mémoire d'Ordinapoche.

ADRESSE CONTENU	ADRESSE CONTENU	ADRESSE CONTENU
10 000	20 000	29 019
11 001	21 010	30 000
12 020	22 011	31 010
13 030	23 030	32 020
14 022	24 013	33 012
15 023	25 014	34 013
16 033	26 015	35 014
17 034	27 034	36 024
18 026	28 017	37 025
19 027		

Pour jouer, écrivez chacun de vos coups sur une carte d'entrée. Ordinapoche répondra en imprimant son coup sur une carte de sortie.

Si vous jouez le premier, placez la puce à la case 00. Si Ordinapoche joue en premier, placez la puce à la case 01. Si Ordinapoche joue en premier, il est imbattable. Vous gagnerez probablement si vous commencez le premier, mais une erreur peut vous faire perdre.

Pour améliorer le jeu

Sachant que le premier joueur gagne toujours enlève de l'intérêt au jeu. Vous vous êtes peut-être senti comme le joueur qui se fait héler par son ami en allant au Casino.

« Où vas-tu ? » demande l'ami.

« Je vais tenter ma chance au Casino. »

« Espèce d'idiot, ne sais-tu pas que le jeu est truqué ? »

« Bien sûr que je le sais », répond le joueur sans hésiter. « Mais qu'est-ce que je peux faire ? C'est le seul jeu en ville. »

Maintenant, nous aimerais qu'il soit possible de battre celui qui commence, sans toutefois diminuer la qualité du jeu d'Ordinapoche. Ordinapoche devrait toujours être programmé pour jouer le meilleur coup possible. Pour augmenter l'élément de chance, il devrait également être programmé de façon à donner à l'adversaire l'occasion de jouer un mauvais coup. De plus, nous aimerais faire travailler Ordinapoche un peu plus, peut-être en lui faisant calculer combien de cailloux restent dans la rangée après chaque coups.

Comment y arriver ? Eh bien, pour commencer, on peut augmenter le nombre de coups gagnants et perdants au-delà de la limite de dix.

Nouveau tableau des coups possibles

COUPS GAGNANTS			COUPS PERDANTS		
1 ► 0	2 ► 0	3 ► 0		2 ► 1	3 ► 1
1 ► 1			1 ► 2	2 ► 2	3 ► 2
			1 ► 3	2 ► 3	
			1 ► 4	2 ► 4	3 ► 3
			1 ► 5	2 ► 5	3 ► 5
			1 ► 6	2 ► 6	3 ► 6
			1 ► 7	2 ► 7	
			1 ► 8	2 ► 8	3 ► 7
			1 ► 9	2 ► 9	3 ► 9
			1 ► 10	2 ► 10	3 ► 10
			1 ► 11	2 ► 11	
			1 ► 12	2 ► 12	3 ► 11
			1 ► 13	2 ► 13	3 ► 12

Ce nouveau tableau nous révèle deux choses intéressantes. D'abord, le fait d'avoir augmenté le nombre initial de cailloux nous permet de commencer le jeu avec un coup perdant. Avec treize cailloux, par exemple, un joueur non averti peut au premier jeu faire l'un ou l'autre des deux mauvais coups suivants : $2 \blacktriangleright 11$ ou $3 \blacktriangleright 10$. Donc, le fait d'augmenter le nombre initial de cailloux satisfait un de nos buts.

La deuxième chose qui nous frappe est la disposition régulière des éléments, qui à première vue ont une forme géométrique.

COUPS GAGNANTS			COUPS PERDANTS		
XXX	XXX	XXX		XXX	XXX
XXX			XXX	XXX	XXX
			XXX	XXX	XXX
XXX	XXX	XXX		XXX	XXX
			XXX	XXX	XXX
XXX	XXX	XXX		XXX	XXX
			XXX	XXX	XXX
XXX	XXX	XXX		XXX	XXX
			XXX	XXX	XXX
XXX	XXX	XXX		XXX	XXX
			XXX	XXX	XXX

Un examen plus approfondi nous révèle que les données ont un cycle de quatre; autrement dit, le fait d'additionner ou de soustraire un multiple de quatre de la rangée de cailloux ne change en rien le résultat d'un jeu. Par exemple, les coups $2 \blacktriangleright 1$, $2 \blacktriangleright 5$, $2 \blacktriangleright 9$, $2 \blacktriangleright 13$, etc. sont tous des coups perdants. La bonne riposte à chacun de ceux-ci est d'enlever un caillou, pour obtenir les coups gagnants $1 \blacktriangleright 0$, $1 \blacktriangleright 4$, $1 \blacktriangleright 8$ et $1 \blacktriangleright 12$.

Nous pouvons nous servir de cette disposition régulière pour programmer Ordinapoche de manière à jouer avec n'importe quel nombre de cailloux jusqu'à concurrence de 999 cailloux. A cette fin, nous ferons jouer Ordinapoche comme s'il était capable de soustraire quatre, un nombre suffisant de fois, afin de réduire la rangée à un nombre plus petit que quatre. Ordinapoche fera alors son jeu comme si la rangée contenait seulement ce dernier nombre de cailloux. En d'autres termes, quel que soit le nombre de cailloux dans la rangée, le jeu d'Ordinapoche sera toujours basé sur un nombre plus petit que quatre. Cette technique éliminera la nécessité de conserver 999 ripostes dans la mémoire d'Ordinapoche. En fait, nous n'aurons besoin que de 16 entrées dans la table de consultation d'Ordinapoche.

Stratégie gagnante

En préparant notre stratégie, nous devons prendre en considération les trois points suivants :

1. Déterminer le coup d'Ordinapoche quand son adversaire joue un coup perdant, laissant un nombre impair de cailloux.

Notre tableau nous montre que des deux ripostes possibles à un coup perdant qui laisse un nombre impair de cailloux, l'une est toujours gagnante et l'autre perdante. Dans tous ces cas, le choix d'Ordinapoche sera toujours le coup gagnant. Par exemple, les deux ripostes possibles au coup perdant $2 \blacktriangleright 11$ sont $3 \blacktriangleright 8$ (gagnante) et $1 \blacktriangleright 10$ (perdante). La riposte d'Ordinapoche sera $3 \blacktriangleright 8$.

2. Déterminer le coup d'Ordinapoche quand son adversaire joue un coup perdant et qu'il reste un nombre pair de cailloux.

Un examen du tableau nous montre que les deux coups possibles dans ce cas sont toujours gagnants. Les seuls cas où il n'y a pas deux coups possibles sont ceux du premier groupe de coups perdants. Cela est attribuable au fait que la plupart de ces coups ne laissent pas assez de cailloux. Un coup tel que $2 \blacktriangleright 1$ permet une seule riposte : $1 \blacktriangleright 0$.

Le premier groupe est donc un cas spécial. Pour simplifier et pour réduire le surmenage de la mémoire d'Ordinapoche, nous déterminerons ses ripostes à partir de ce groupe spécial : à cause de la disposition régulière des éléments du tableau, nous pouvons utiliser les mêmes choix pour tous les groupes subséquents; par exemple, le seul coup possible en réponse au coup perdant $1 \blacktriangleright 2$ est $2 \blacktriangleright 0$. Alors, la riposte d'Ordinapoche, à tous les coups périodiquement équivalents tels que $1 \blacktriangleright 6$, $1 \blacktriangleright 10$, $1 \blacktriangleright 14$, etc., sera d'enlever deux cailloux.

3. Placer l'adversaire d'Ordinapoche dans une position où il a le maximum de chances de jouer un mauvais coup.

Supposons que l'adversaire d'Ordinapoche joue un coup gagnant; la riposte d'Ordinapoche devrait lui laisser autant de chances que possible de jouer un coup perdant au prochain tour. Si le coup du joueur est $1 \blacktriangleright 8$, la riposte sera soit $2 \blacktriangleright 6$, soit $3 \blacktriangleright 5$. Si nous choisissons $2 \blacktriangleright 6$, le choix subséquent du joueur doit être un coup gagnant, peu importe s'il choisit les coups $1 \blacktriangleright 5$ ou $3 \blacktriangleright 3$. Par ailleurs, si notre riposte est $3 \blacktriangleright 5$, le joueur a une chance de jouer un coup perdant ($2 \blacktriangleright 3$) aussi bien qu'un coup gagnant ($1 \blacktriangleright 4$). Donc, $3 \blacktriangleright 5$ sera notre meilleur choix. Dans

la mesure du possible, nous procéderons de la même façon pour répliquer aux coups gagnants de l'adversaire.

Tableau des coups et ripostes d'Ordinapoche

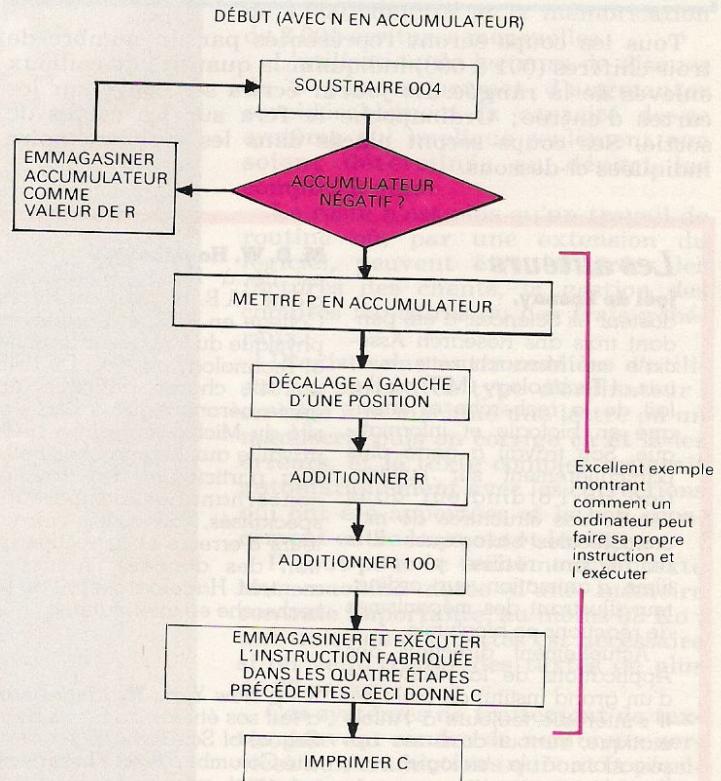
Une fois notre stratégie établie, nous pouvons déterminer quelle sera la riposte d'Ordinapoche à n'importe quel coup. En ajoutant ces ripostes entre parenthèses à notre tableau, ce dernier sera plus complet et plus instructif.

COUPS GAGNANTS	COUPS PERDANTS
$1 \blacktriangleright 0 (*)$	$2 \blacktriangleright 1 (1)$
$2 \blacktriangleright 0 (*)$	$3 \blacktriangleright 1 (1)$
$1 \blacktriangleright 1 (*)$	$1 \blacktriangleright 2 (2)$
	$2 \blacktriangleright 2 (1)$
	$3 \blacktriangleright 2 (2)$
	$1 \blacktriangleright 3 (3)$
	$2 \blacktriangleright 3 (3)$
$3 \blacktriangleright 3 (2)$	
$1 \blacktriangleright 4 (3)$	$2 \blacktriangleright 5 (1)$
$2 \blacktriangleright 4 (1)$	$3 \blacktriangleright 5 (1)$
$3 \blacktriangleright 4 (1)$	$1 \blacktriangleright 6 (2)$
$1 \blacktriangleright 5 (2)$	$2 \blacktriangleright 6 (1)$
	$3 \blacktriangleright 6 (2)$
$3 \blacktriangleright 7 (2)$	$1 \blacktriangleright 7 (3)$
$1 \blacktriangleright 8 (3)$	$2 \blacktriangleright 7 (3)$
$2 \blacktriangleright 8 (1)$	
$3 \blacktriangleright 8 (1)$	$2 \blacktriangleright 9 (1)$
$1 \blacktriangleright 9 (2)$	$3 \blacktriangleright 10 (2)$
	$1 \blacktriangleright 10 (2)$
$3 \blacktriangleright 11 (2)$	$2 \blacktriangleright 10 (1)$
$1 \blacktriangleright 12 (3)$	$3 \blacktriangleright 11 (3)$
$2 \blacktriangleright 12 (1)$	
$3 \blacktriangleright 12 (1)$	
$1 \blacktriangleright 13 (2)$	

* Fin du jeu.

De même, la première « mise à jour de N » exige simplement l'utilisation d'une instruction CLA qui place « N » en accumulateur, puis d'une instruction SUB qui soustrait « P » de « N »; et finalement une instruction STO pour emmagasiner « N » mis à jour dans une case définie.

Cependant, la méthode d'utilisation des deux instructions « Trouver R » et « Trouver C » n'est ni simple ni évidente. Donc,, un ordinogramme détaillé de ces étapes est nécessaire (micro-ordinogramme).



Micro-ordinogramme pour trouver R et C.

Programme et directives

Ces ordinogrammes terminés, il nous reste donc à les codifier mécaniquement en instructions d'Ordinapoch et à choisir les cases appropriées pour les placer. De même, comme notre stratégie nous a indiqué les meilleures ripostes d'Ordinapoch, il suffit de les placer en mémoire dans d'autres cases.

Programme numéro 9 : jeu de Nim amélioré

Si le joueur doit débuter, placer la puce à la case 52.
Si c'est Ordinapoch, placer la puce à la case 53.

ADRESSE	CONTENU	COMMENTAIRES
52	015	Lire P
53	114	
54	715	Mise à jour de N
55	614	
56	514	Imprimer N
57	718	Soustraire 4
58	361	Acc. négatif ?
59	617	Emmagasiner
60	857	Saut
61	115	Charger Pen acc. (00P en acc.)
62	410	Décalage vers la gauche (0P0 en acc.)
63	217	Additionner R (0PR en acc.)
64	219	Additionner 100(1PR en acc.)
65	666	Emmagasiner (1PR en case 66)
66	100	Charger C (CLA contenu de la case PR)
67	616	Emmagasiner C
68	516	Imprimer C
69	114	
70	716	Mise à jour de N
71	614	
72	514	Imprimer N
73	952	Stop (retour à 52)

Tous les coups seront représentés par un nombre de trois chiffres (001 à 003) indiquant la quantité de cailloux enlevés de la rangée. Le joueur écrira ses coups sur les cartes d'entrée; Ordinapoch le fera sur les cartes de sortie. Ses coups seront placés dans les cases-mémoire indiquées ci-dessous.

Les auteurs

Joël de Rosnay,

docteur ès Sciences, a été pendant trois ans Research Associate au Massachusetts Institute of Technology (MIT) où il a fait de la recherche et enseigné en biologie et informatique. Son travail a porté plus particulièrement sur l'utilisation de l'ordinateur dans l'étude des structures de macromolécules biologiques. Il a notamment réalisé plusieurs films d'animation sur ordinateur illustrant des mécanismes de réactions en biochimie.

Actuellement directeur des Applications de la Recherche d'un grand institut de biologie, il s'intéresse toujours à l'informatique, surtout dans ses applications à la biologie et à l'enseignement.

M. D. W. Hagelbarger

est né à Kipton (Ohio). Diplômé (A.B. degree) du Hiram College en 1942, et Docteur en physique du California Institute of Technology en 1947. De 1946 à 1949, chargé de cours en génie aéronautique à l'université du Michigan. Depuis 1949, travaille aux Laboratoires Bell ; il a participé à des travaux concernant les ordinateurs spécialisés, les codes correcteurs d'erreurs et la récupération des données. Actuellement, M. Hagelbarger fait de la recherche en informatique.

Né à New York, **M. Fingerman** a fait ses études au Bronx High School of Science et à l'université Columbia. **Saul Fingerman** est entré au service des Rela-

Tableau des coups d'Ordinapoch

ADRESSE	CONTENU	ADRESSE	CONTENU
00	001	20	001
01	001	21	001
02	002	22	001
03	003	23	003
10	003	30	001
11	002	31	001
12	002	32	002
13	003	33	002

Tableau des constantes et des variables

ADRESSE	CONTENU	DESCRIPTION
14	N (début)	N = nombre de cailloux dans la rangée
15	000	P = nombre de cailloux enlevés par le joueur
16	000	C = nombre de cailloux enlevés par Ordinapoch
17	N (début)	R = résidu
18	004	La valeur 4 à soustraire
19	100	Code d'opération numéro 1, CLA

Directives

- Placez le programme et le tableau des coups d'Ordinapoch en mémoire.
- Placez le tableau des constantes et variables en mémoire. Notez que la valeur initiale N est placée dans la case 17 réservée au résidu, ainsi que dans la case 14. Ceci permet au programme de fonctionner même pour les valeurs initiales de N plus petites que quatre.
- Commencez en plaçant la puce à la case 53, si Ordinapoch doit commencer le premier. Si le joueur commence le premier, placez la puce à la case 52. Le joueur devra écrire le nombre de cailloux qu'il enlève (001, 002 ou 003) sur les cartes d'entrée.

NOTE : si tout ceci vous a paru bien long et compliqué pour un programme aussi court, nous ne pouvons que dire que c'est là la façon normale de procéder en programmation. Le travail consiste généralement à analyser le problème, à trouver une façon de le résoudre et à le décomposer en éléments compatibles avec l'ordinateur.

Vous avez maintenant une idée assez exacte de ce qu'est la programmation d'un ordinateur.

tions publiques et des publications des Laboratoires Bell en 1964, après avoir servi pendant plusieurs années dans la marine marchande comme radio. Il a conçu et mis au point plusieurs supports pédagogiques, notamment le Robot... pensant.

Quelques livres à lire

ÉDUCATION ET INFORMATISATION DE LA SOCIÉTÉ

J.C. Simon (Doc. Français, 1980).

LE FIL D'ARIANE

Jean-Pierre Bouhot. 2 volumes (Ed. de l'Informatique), 1980.

LES MICROPROCESSEURS

Rodney Zaks-Pierre Le Beux (SYBEX), 1979.

INTRODUCTION AUX MI-

CROPROCESSEURS ET MICRO-ORDINATEURS.

H. Lilien (Ed. RADIO).

ENTRAÎNEMENT À LA PROGRAMMATION.

Warnier-Flanagan (Ed. d'Organisation).

INITIATION À LA PROGRAMMATION EN FORTRAN.

J.B. Krin (Cours ENSTA).

FINITE AND INFINITE MACHINE.

Marvin Minsky (Prentice Hall).

LE LANGAGE DE PROGRAMMATION PASCAL.

Ph. Kruchter (Ed. Eyrolles).

LA 3^e VAGUE.

A. TOFLER (Ed. Denoël), 1980.

VOTRE PREMIER ORDINATEUR.

Rodney Zaks (SYBEX), 1981.

INTRODUCTION AU BASIC SUR MICRO-ORDINATEUR.

Pierre Le Beux (SYBEX), 1979.

LA RÉVOLUTION INFORMATIQUE

Et l'homme crée la puce

- Depuis le premier ENIAC, un géant de 19 000 tubes électroniques créé en 1946 — aujourd'hui pièce de musée — aux circuits intégrés inventés en 1959, l'itinéraire technologique de l'homme est comparable au passage de l'âge de la pierre taillée à celui de la machine-outil.

page 1

L'intelligence artificielle

- Françoise Harrois-Monin, notre correspondante aux Etats-Unis, réside dans la « Silicon Valley », le sanctuaire de l'informatique. C'est de là que nous viennent toutes les inventions liées aux ordinateurs. Vous apprendrez bien des choses étonnantes dans son article sur « l'intelligence des machines ».

page 8

Mieux vivre avec l'informatique

- Les progrès réalisés vers l'infiniment petit nous réservent encore bien des surprises. Dans un style vivant, N. Zaoui nous entraîne dans le monde informatisé de demain : écoles, monnaies, voitures, informations à domicile, etc.

page 10

L'ORDINAPOCHE

- C'est un véritable simulateur d'ordinateur. C'est-à-dire qu'en deux ou trois heures, vous apprendrez comment fonctionne un vrai ordinateur, qu'il soit grand ou petit, et vous saurez écrire votre premier programme. C'est l'instrument indispensable des années 1980 si vous tenez à ne pas rater la Révolution informatique.

page 21

Le banc d'essai des micro-ordinateurs

- Après avoir lu cet article clair, complet et détaillé, vous serez en mesure de choisir et de sélectionner vous-même le micro-ordinateur indispensable à votre travail, que vous soyez médecin, enseignant, étudiant, avocat, architecte, patron de PME, directeur d'agence ou chef de service. Vous pourrez aussi connaître son prix.

page 45

Votre avenir serait-il dans l'informatique ?

- Cinq cent mille postes à pourvoir dans les années à venir. Albert Ducrocq, grand spécialiste des problèmes techniques et scientifiques, a rédigé ce chapitre important pour l'avenir des jeunes en France. Il passe en revue tous les métiers de l'informatique et explique les filières à suivre pour chacun d'entre eux. Vous saurez tout sur les métiers d'avenir, les écoles et les salaires actuels.

page 54

Le petit dictionnaire de l'informatique

- Soixante-dix mots clés pour comprendre le langage de l'informatique.

page 61