initiation

Laissez vous aller à cuisiner pour apprendre à programmer

Apprendre à programmer ne paraît pas toujours très simple quand on débute. Quelle démarche faut-il suivre et que signifient tous ces codes divers et variés ? Cet article vous en donne des exemples : finalement, la programmation est une drôle de cuisine !

■ Dans le précédent numéro, après un bref aperçu sur l'utilisation d'une calculatrice algébrique en mode calcul, nous avons réalisé un premier petit programme. Vous vous doutiez que nous n'en resterions pas là...

Et pourtant, nous avons déjà vu l'essentiel : les programmes plus « sophistiqués » que l'on peut faire exécuter à une calculatrice programmable ne sont pas beaucoup plus compliqués que le premier.

Notre programme de calcul de la longueur d'un toit ne faisait que reproduire l'ordre d'exécution d'un calcul au clavier : c'est ce que j'appelle de la programmation « en ligne ». Les programmes de ce genre sont, bien entendu, les plus faciles à réaliser et certainement les plus couramment utilisés par les informaticiens amateurs, du moins au début de leur apprentissage. Mais il est possible de faire travailler la calculatrice davantage. Et c'est ce que nous allons étudier aujourd'hui.

Nous examinerons d'abord une étape qui, parce qu'elle est trop souvent négligée, est à l'origine de bien des déboires : la conception d'un programme.

Lorsque vous préparez un bon petit plat, par quoi commencez-vous ? Allumer le fourneau ? Certainement pas ! Vous cherchez plutôt quelle est la recette que vous allez pouvoir exécuter. Pour cela vous vous demandez « Qu'est-ce que j'aimerais bien offrir à mes invités ? » et vous imaginez un résultat, vous en salivez peut-être déjà... Appelons cette étape « recherche de finalité ». Ensuite, vous étudiez les grandes lignes de la réalisation, mais sans entrer vraiment dans le détail de la recette. Si par exemple vous désirez faire des « îles flottantes », vous remarquez qu'il faut d'une part monter des blancs d'œuf en neige et d'autre part préparer une crème anglaise. Il faudra donc des œufs, du lait, etc... Cette étape peut être nommée « recherche des moyens ». Par la suite, il faudra aller dans le détail de la recette, calculer les proportions en fonction du nombre de convives prévus, établir la succession des opérations ; c'est la partie « conception » de votre plat. Il ne restera plus qu'à réaliser, c'est-à-dire à suivre tout le processus de la recette jusqu'au résultat final qui doit correspondre (si vous êtes bon cuisinier) à ce que vous espériez. Cela peut s'appeler la « réalisation ». Dans le courant de la réalisation, on peut être amené à goûter le plat, à y ajouter du sucre, du sel. C'est « l'adaptation ».

Je vous rassure tout de suite, vous êtes bien en train de lire « l'Ordinateur de poche ». Si nous avons parlé cuisine, c'est pour dégager les analogies existant entre une recette et un programme. Les quatre étapes que nous avons remarquées sont en effet les mêmes, à peu de choses près, dans les deux cas.

Où désire-t-on arriver ?

On peut aussi appeler cette étape définition d'objectif, cahier des charges, ou énoncé du problème : pendant cette phase, le programmeur va définir ce qu'il veut obtenir en fin de calcul. La nécessité de cette étape semble vraiment évidente, mais on laisse trop souvent de côté cette opération pour partir à l'aveuglette. Je vous conseille de prendre dès le départ une bonne habitude : définissez toujours clairement votre but, exactement comme si vous deviez faire réaliser le programme par quelqu'un d'autre. Ecrivez les grandes lignes de l'énoncé du problème comme vous le feriez pour un exercice de maths ou de physique.

Nous allons détailler les étapes de la réalisation d'un programme autour d'un exemple : quand nous faisons nos courses dans un supermarché, il nous est souvent difficile de comparer les prix d'articles similaires à cause de la variété des conditionnements, les différences de poids obligent à faire des règles de trois. Nous allons donc écrire un programme qui nous donnera le prix des articles au kilo, quelle que soit la contenance de l'emballage. Voilà pour l'énoncé du problème. A partir

Cuisiner pour apprendre à programmer

de cet énoncé, nous pouvons envisager d'aller plus loin. Nous savons maintenant quelle est notre cible. Reste à trouver le moyen de l'atteindre.

Passons à la recherche des moyens

Cette étape peut s'envisager de deux façons : matériel et logiciel. En ce qui concerne le matériel, pas de problème pour nous : il s'agit d'un ordinateur de poche ou d'une calculatrice programmable. En ce qui concerne le logiciel, si nous laissons de côté le choix du langage de programmation qui ne se pose pas à nous, la recherche des moyens va nous contraindre à définir les types de calculs qui seront requis pour parvenir au but. Il sera nécessaire également de préciser quels éléments, quelles données seront disponibles. Si vous avez décidé de faire des îles flottantes et si vous n'avez pas d'œufs, la situation est sans espoir !

Poursuivons notre exemple de comparaison de prix dans un supermarché. La recherche des moyens nous conduira à préciser les données dont nous disposons et le type de calcul utilisable.

Données : - poids d'un article
- prix de cet article

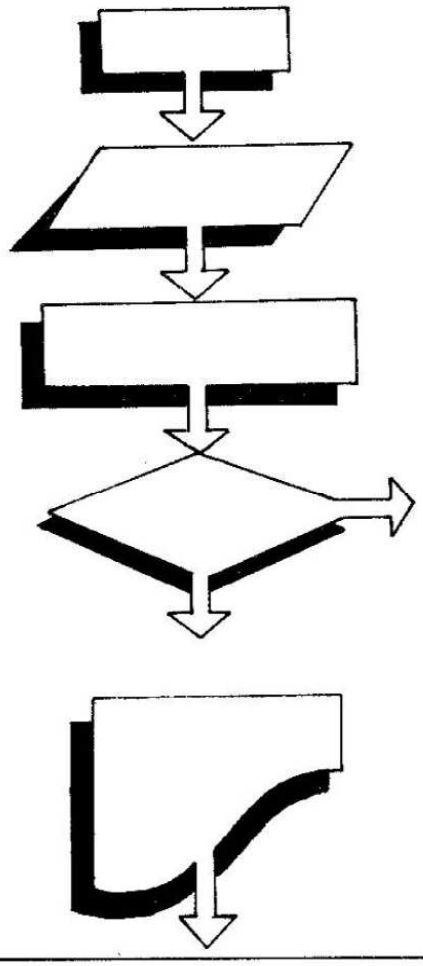
Type de calcul : ce sera ici une « règle de trois » : (prix à l'article) multiplié par (poids normalisé), et divisé par (poids de l'article).

Dans le jargon de l'informatique, cette phase de la conception d'un programme s'appelle un algorithme. L'algorithme est au programme ce que le synopsis (le résumé du scénario) est au film. Seules les grandes lignes y sont tracées de manière à donner une vision synthétique de l'ensemble.

Pour plus de commodité, on a coutume de représenter cet algorithme de manière standardisée. Cette représentation, quand elle prend la forme d'un dessin, s'appelle un organigramme. Je ne sais plus qui disait (et j'avoue ne pas avoir le courage de chercher) qu'un dessin vaut mille mots. C'est très vrai en tout cas à propos de l'organigramme.

Les symboles utilisés sont nombreux (cf. figure 1). A mon avis, il est préférable de se limiter au strict minimum. Cela nous fait en tout six symboles à retenir : ils nous permettront de représenter pratiquement tous les cas possibles.

Figure 1



Instruction de début et de fin de programme

Entrée de données

Traitement (calcul)

Test, comparaison, aiguillage dans la conduite d'un calcul. (Nous verrons plus tard comment cela marche).

Sortie de données, visualisation de résultats.

Figure 2

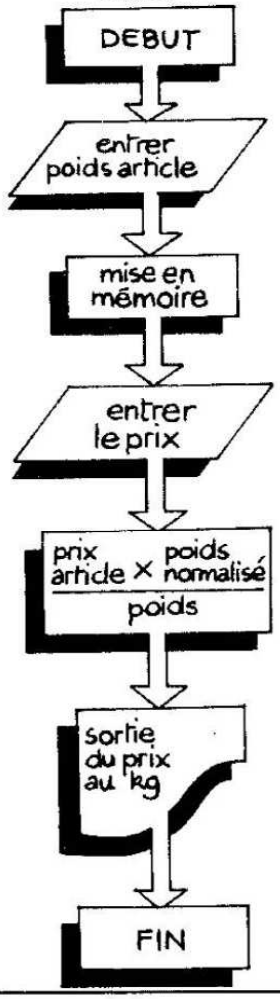


Tableau 1

Opération	Programme	Commentaire
Entrée du poids de l'article (unité prévue : gramme)	STO 0 R/S	Mise en mémoire Arrêt pour entrée de la 2e donnée
Le prix de l'article est entré dans le registre d'affichage puis divisé par la première donnée	÷ RCL 0	Rappel du poids ou de la contenance qui est le diviseur
et multiplié par 1 000	X 1 000	Nous avons entré un poids en grammes et nous voulons un résultat en kg
	= R/S	Arrêt pour regarder le résultat : prix au kg.

Notre exemple de calcul de prix au kilo donnera l'organigramme de la figure 2.

Grâce à ce schéma, il sera possible à n'importe quel programmeur, indépendamment du type de matériel utilisé, de comprendre la succession des opérations. L'organigramme doit être clair : il doit donc rester simple. Il ne saurait être question d'y exposer en détail toutes les opérations qui seront utilisées dans les calculs. Souvenez-vous que l'organigramme représente une vision globale du programme. C'est une carte au 1/1 000 000e, ce n'est pas une carte d'état-major : seules les autoroutes et les nationales y figurent.

Puis vient le moment de la conception

C'est maintenant que nous allons rentrer dans les détails et rédiger le programme proprement dit. Autrement dit, nous allons prendre une feuille de papier et y écrire les instructions de programmation en suivant le schéma de l'organigramme.

Dans le cas de programmes compliqués, on aura intérêt à fractionner au maximum cette écriture et à considérer les différentes étapes comme des éléments séparés et autonomes. Pour reprendre l'analogie avec nos « îles flottantes », nous faisons d'une part les œufs en neige et d'autre part la crème anglaise.

Dans la pratique de l'écriture, on peut séparer les différentes opérations en pavés distincts qui pourront être essayés les uns après les autres. En procédant de la sorte, on facilitera beaucoup la mise au point du programme et les éventuelles modifications.

Nous pouvons écrire ainsi notre programme de calcul de prix comme dans le tableau 1.

Réalisation et mise au point

Nous pouvons maintenant sortir la calculatrice de son étui et commencer à pianoter. Tous les éléments de notre recette étant réunis, nous pouvons mettre au four.

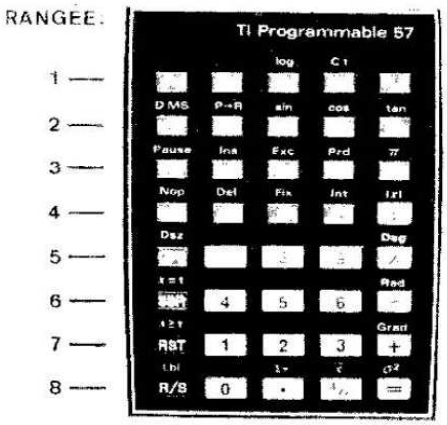
Allumer la calculatrice, appuyer sur LRN pour passer en mode de programmation, puis introduire les instructions. Repasser en mode calcul en appuyant sur LRN à nouveau.

Si tout ce qui précède a été soigneusement réalisé, le programme doit « tourner » du premier coup. Mais il est préférable de s'en assurer en exécutant un premier calcul qui aura été au préalable réalisé manuellement.

Nous allons par exemple calculer le prix au kilo d'un article qui coûte 18 F et dont l'emballage contient 500 g. Si tout va bien, nous devons obtenir 36 F.

Sur la calculatrice faire RST pour ramener le pointeur de programme au départ (pas 00) puis introduire 500 et appuyer sur R/S. Entrer ensuite 18 et relancer avec R/S. Notre affichage indique 360 (le votre affiche certainement une valeur différente, mais continuez à lire sans vous en inquiéter !). Bizarre ! Y aurait-il une erreur ?

Nous allons d'abord contrôler que le programme a été convenablement introduit. Pour cela nous pouvons le « lister » pas à pas c'est-à-dire dérouler toutes les instructions et opérations qui ont été programmées. L'ennui, c'est que l'affichage ne peut donner que des chiffres, alors comment reconnaître les touches qui ont été mises en mémoire ? C'est simple : chaque touche est numérotée. Et pour que cette numérotation soit simple à retrouver, le constructeur a prévu une organisation logique. Les chiffres de 0 à 9 sont représentés par leur propre valeur (de 00 à 09). Les autres touches sont numérotées selon leur emplacement sur le clavier. Une touche peut être ainsi définie par le numéro de la rangée où elle se trouve suivi par celui de sa colonne.



Les deux touches 2nd et INV servent à étendre le clavier, sans augmenter le nombre de touches. Le repérage d'une instruction appelée après la touche 2nd se fait en ajoutant 5 au numéro de sa colonne, sauf pour les touches de la dernière colonne auxquelles on retire 5.

La touche INV est indiquée par un signe — devant le code de la touche.

Exécutons ensemble la liste du programme qui est en mémoire : Ramener le pointeur au départ : RST LRN passage en mode programme. L'affichage indique :

numéro de pas de programme	00 ↑	32 ↑ code de la touche STO	0 ↑ identification (ici numéro de la mémoire concernée)

Cuisiner pour apprendre à programmer

Tableau 2

Numéro des pas	Code des touches	Touches	Commentaires
00	32 1	STO 1	Mise en mémoire du prix
01	81	R/S	Arrêt pour entrer le 2e nombre
02	32 0	STO 0	Mise en mémoire du poids
03	33 1	RCL 1	Rappel du prix
04	45	÷	Diviser
05	33 0	RCL 0	Rappel du poids
06	55	×	Mutiplier
07	01	1	Par 1 000
08	00	0	
09	00	0	
10	00	0	
11	85	=	Résultat
12	81	R/S	Arrêt d'exécution

Tableau 3

Numéro des pas	Code des touches	Touches	Commentaires
00	32 1	STO 1	Mise en mémoire du prix
01	81	R/S	Arrêt pour entrer les poids
02	25	1/x	Inverse
03	55	×	Multiplier
04	33 1	RCL 1	Rappel prix
05	55	×	Multiplier
06	01	1	par
07	00	0	1 000
08	00	0	
09	00	0	
10	85	=	Résultat
11	81	R/S	Arrêt d'exécution

Tableau 4

Numéro des pas	Code des touches	Touches	Commentaires
00	32 1	STO 1	Mise en mémoire du prix
01	81	R/S	Arrêt, entrée du poids
02	−39 1	INV 2nd Prd 1	Diviser le prix en mémoire
03	01	1	1 000
04	00	0	
05	00	0	
06	00	0	
07	39 1	2nd Prd 1	Mutiplier en mémoire
08	33 1	RCL 1	Rappel du résultat
09	81	R/S	Arrêt d'exécution

Tableau 5

Numéro des pas	Code des touches	Touches	Commentaires
00	45	÷	Entrée du prix d'un article, diviser
01	81	R/S	Entrée du poids en gr
02	55	×	Multiplier
03	03	3	par 10^3
04	−18	INV 2nd Log	(3 INV 2nd Log = 1 000)
05	85	=	Résultat
06	81	R/S	Arrêt d'exécution

La touche STO que nous avions appuyée est placée sur la troisième rangée, deuxième colonne. Le 0 qui est derrière est un identificateur, il représente ici le numéro de la mémoire où doit être conservé le nombre à l'affichage. Sur d'autres calculatrices, le numéro est parfois enregistré au pas suivant.

Pour passer au pas suivant, il faut appuyer sur la touche SST (*Single STep,* un seul pas). L'affichage donne :

01	81
↑	↑
numéro de pas	code de la touche R/S

81 = huitième rangée, première colonne. Vous pouvez vérifier, c'est bien R/S qui y est placé.

SST	02	45		(÷)
SST	03	33	0	(RCLO)
SST	04	55		(X)
SST	05	01		(chiffre 1)
SST	06	00		(chiffre 0)
SST	07	00		"
SST	08	00		"
SST	09	00		"
SST	10	85		(=)
SST	11	81		(R/S)

C'est là que l'on peut retrouver l'erreur qui s'est glissée dans notre programme. Il y a un zéro de trop au pas 09. Nous avions écrit 10 000 au lieu de 1 000. Il faut donc supprimer ce pas. Pour cela, on se placera au pas 09, en utilisant SST ou au besoin la touche de retour arrière BST (*Back Step,* un pas en arrière) qui permet de dérouler le programme dans l'autre sens, vers le début. Vous y êtes ? Bien. Faites maintenant 2nd Del (touche 47). L'affichage donne 09 85 : le pas 10 est remonté d'un cran et le 0 fautif a été supprimé. Parfait !

Nous aurions pu tout aussi bien corriger le programme en remplaçant le 0 du pas 09 par 2nd Nop (code 46) (*no operation,* pas d'opération). Cette instruction n'a pas d'action sur l'exécution du pro-

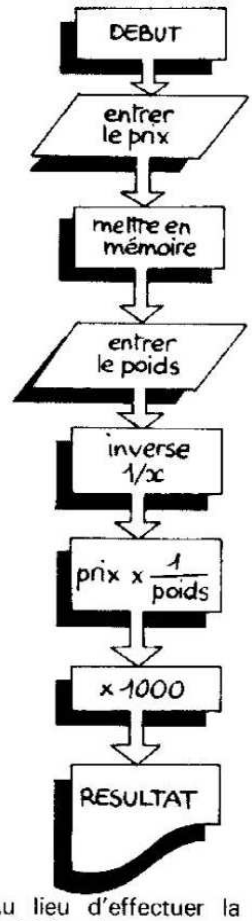
gramme mais elle permet de supprimer des pas sans modifier les pas suivants.

Si nous avions au contraire oublié une instruction, nous aurions pu la rajouter après coup sans réécrire tout le programme : il aurait suffi de se placer sur le pas suivant celui que l'on a oublié (avec SST ou BST), de fare 2nd Ins (*Insert,* insérer) (code 37) et d'inscrire l'instruction oubliée. Quand on utilise 2nd Ins, le pointeur reste au même numéro de pas, mais tout le reste du programme est décalé vers le bas. Toutes ces opérations d'édition et de correction de programmes sont longues à décrire mais elles s'exécutent très simplement. Avec un peu de pratique, on les assimile très vite.

Il ne reste plus qu'à s'en servir

Quand vous êtes certain que votre programme fonctionne correctement, il ne vous reste plus qu'à l'utiliser. C'est la dégustation du plat que vous avez cuisiné avec tant de soin. Bon appétit !

Figure 3



Au lieu d'effectuer la division, on multiplie le prix par l'inverse du poids. Cela revient au même mais nous économisons une mémoire.

Dans l'exemple de calcul de prix que nous avons suivi, après chaque affichage de résultat, vous faites RST puis vous introduisez un nouveau poids, R/S, un nouveau prix, R/S et vous lisez le nouveau résultat.

Exercice complémentaire : étudiez le même programme pour qu'il fonctionne en entrant en premier le prix d'un article puis son poids. Réfléchissez un moment avant de lire la suite.

Facile, direz-vous, il suffit d'utiliser une autre mémoire pour conserver le prix et de faire la division entre le contenu des deux mémoires. Cela donne le tableau 2.

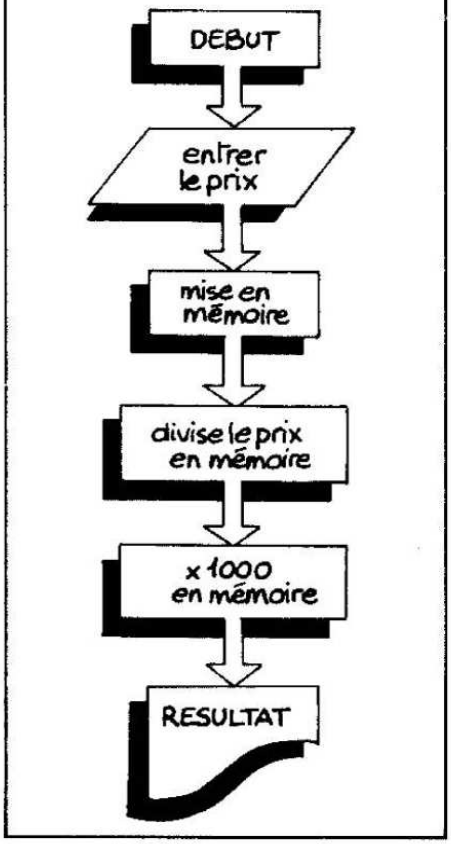
C'est effectivement une méthode pour obtenir le même résultat, mais il y en a d'autres qui permettront d'économiser une mémoire et des pas de programme. Il faut revenir en arrière, choisir un autre algorithme et dessiner un nouvel organigramme. Le programme devient alors celui du tableau 3.

Nous pouvons essayer une troisième méthode qui, tout en conservant l'avantage de l'économie de mémoire, fera gagner encore des pas de programme. Cette fois-ci, nous utiliserons les opérations directes en mémoire. L'algorithme est encore différent, et l'organigramme qui le représente devient celui de la figure 4 et le programme est présenté dans le tableau 4.

Cet algorithme nous a permis de n'utiliser qu'une mémoire et de gagner encore des pas de programmes. Nous n'employons plus que 10 pas contre 11 dans la première version (avec entrée du poids en premier). Mais on peut faire encore mieux : ne plus utiliser de mémoire, en se servant directement des deux registres de travail de la calculatrice ; ce sont 2 mémoires qu'on appelle traditionnellement X et Y qui stockent les deux opérandes d'un calcul.

La multiplication par 1 000 peut aussi (sur notre machine) être raccourcie en utilisant les antilogarithmes décimaux (puissances de 10) : × 1 000 est équivalent à × 3, INV 2nd Log : ce qui prenait 5 pas n'en occupe plus que 3. Le tableau 5 vous donne le programme ainsi optimisé : seulement 7 pas de programme, et plus aucun registre de mémoire utilisé, de mieux en mieux...

Figure 4



Le gain obtenu sur l'occupation des mémoires (mémoires de programme et de données) présente peu d'intérêt ici. Mais dans le cas de programmes plus longs, en gagnant un pas par-ci, un autre par-là, on arrive à faire tenir des calculs complexes dans nos 50 cases de mémoire programme.

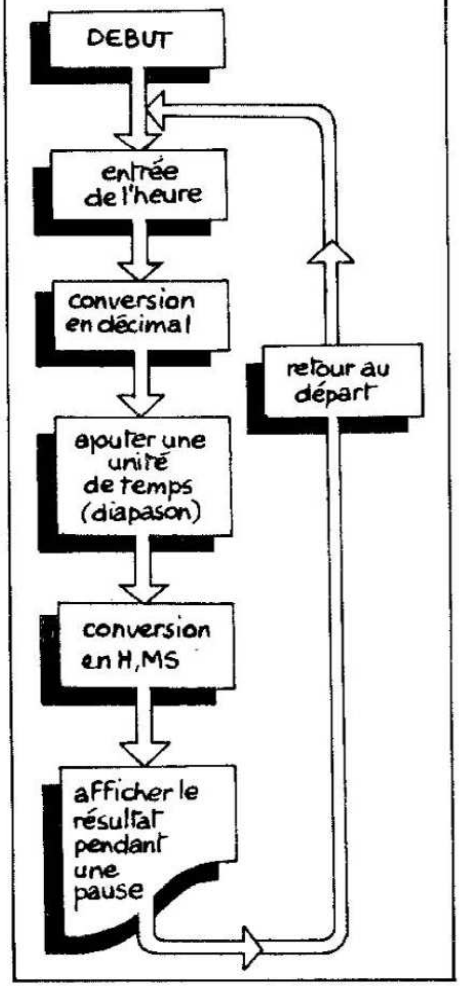
Cet exercice nous a permis de constater que, même sur un programme rudimentaire, il y a souvent une optimisation possible. Mais l'amélioration ne consiste pas seulement à enlever les pas inutiles (il y en a parfois) : il faut aussi rechercher si l'algorithme utilisé est bien adapté aux possibilités de la calculatrice. C'est pourquoi j'insiste sur l'importance de la préparation des programmes :

— bien définir le but du voyage
— chercher le chemin le plus court
— ne pas hésiter à réfléchir sur le parcours envisagé pour voir s'il n'existe pas de raccourci.

Ce point de vue est rarement développé dans les notices des calculatrices programmables et le débutant se laisse facilement aller à faire de la programmation à « la volée » sans prendre le temps de réfléchir et d'écrire sur un papier. Bonne calculatrice, mais peut mieux faire !

Cuisiner pour apprendre à programmer

Figure 5



Si l'écriture d'un programme se bornait à la résolution d'expressions arithmétiques, ce ne serait peut-être déjà pas si mal. Mais il existe de nombreux cas où l'on peut confier davantage de travail à la calculatrice.

Elle ne rechigera pas à faire des tests ou des calculs itératifs, autrement dit des calculs dans lesquels le résultat d'une opération est utilisé comme donnée de la même opération qui est exécutée en boucle.

RST au travail dans un programme

Jusqu'ici, nous n'avons utilisé l'instruction RST que pour ramener le pointeur au départ avant de faire « courir » un programme. Mais RST peut aussi se trouver à l'intérieur du programme. Quand le pointeur rencontrera cette « remise en place », il retournera au départ et recommencera les calculs en prenant comme nouvelle donnée le nombre présent à l'affichage.

Vous connaissez le programme

+
1
=
RST

Il fait compter la machine toute seule. Au premier tour si l'affichage est à zéro, il fait 0 + 1 = 1 puis le pointeur retourne au départ et fait 1 + 1 = 2 et ainsi de suite jusqu'à ce que l'affichage n'en puisse plus ou que les batteries soient déchargées. Un programme qui se mord la queue, en quelque sorte !

Admettons que ce n'est pas très spectaculaire. Mais il est possible de développer des programmes autour de ce principe.

Imaginons de transformer notre calculatrice en chronomètre ou en montre. Nous allons pour cela utiliser la conversion de degrés décimaux en degrés, minutes et secondes, et inversement. L'organigramme du programme pourrait se représenter suivant la figure 5, ce qui va se traduire en langage TI 57 par le programme du tableau 6.

Pour utiliser ce programme, entrez-le en mode LRN, repassez en mode calcul puis entrez l'heure ou faites CLR pour partir à 0 (chronométrage). Puis appuyez RST et R/S pour lancer le programme. Vous remarquez un défilement rapide de chiffres sur l'affichage. Puis l'heure apparaît et reste un peu moins d'une seconde. C'est l'instruction 2nd Pause (code 36) qui permet cet arrêt momentané d'exécution pour la consultation de l'affichage.

Cette horloge n'est pas très précise. Il faut adapter le « nombre diapason » 0,0006943 à votre calculatrice personnelle. De plus, la vitesse de calcul dépendant un peu de la charge des accus, la précision y sera subordonnée.

J'espère cependant que ce petit programme vous donnera d'autres idées pour employer la touche RST. Elle est un premier pas vers une programmation plus évoluée et fournit à la calculatrice des données pour poursuivre toute seule des calculs.

Une prochaine fois, nous étudierons des méthodes pour contrôler le nombre de boucles qu'effectue le programme. De plus en plus perfectionné...

□ Xavier de La Tullaye

Tableau 6

Numéro des pas	Code des touches	Touches	Commentaires
00	26	2nd D.MS	Conversion de l'heure entrée en décimal
01	48 4	2nd Fix 4	Limitation de l'affichage à 4 chiffres
02	75	+	Addition de l'unitée de temps
03	83	•	(battement du diapason)
04	00	0	Ce nombre est ajusté en fonction
05	00	0	de la vitesse de calcul de votre
06	00	0	TI57 : elles ne tournent pas
07	06	6	toutes à la même vitesse
08	09	9	
09	04	4	
10	03	3	
11	46	2nd Nop	Ajustement fin de l'horloge, on
12	46	2nd Nop	peut en ajouter ou en enlever
13	85	. =	Résultat décimal
14	26	INV 2nd D.MS	Conversion en Heures, Minutes, Secondes
15	36	2nd Pause	Affichages de l'heure et poursuite du programme
16	71	RST	Retour au départ