la programmation : les boucles et les tests

Quand un programme se complique un peu, il faut y poser des jalons. Ce sont les étiquettes.

■ Vous savez sans doute que pour pêcher les crabes il y a deux techniques. La première consiste à aller à marée basse entre les rochers avec un crochet : on attrape les bêtes en soulevant les pierres, il faut donc être sportif et ne pas trop compter sur cette méthode pour faire bombance. La seconde technique demande beaucoup plus de matériel : des casiers, un bateau pour aller les mouiller aux endroits propices et des bouées pour repérer où ont été immergés les casiers. (Rassurez-vous, c'est bien *l'Ordinateur de poche* que vous êtes en train de lire...).

Ecrire un programme pour une TI57 ou pour une autre machine présente de grandes similitudes avec cette pêche : ou le programme est simple et il ne demande que peu de choses ou bien il est compliqué et il faut mettre en œuvre de plus gros moyens. Jusqu'ici, nous avons réalisé des programmes simples, et la méthode suivie ressemblait à la pêche au crochet : résultats immédiats, pas de complications, mais peu d'espoir de faire des prises extraordinaires. Préparons maintenant nos casiers pour une pêche plus substantielle.

Lorsque vous mouillez une nasse, l'essentiel de la pêche va se dérouler au fond de l'eau, hors de votre vue. Cette masse est reliée à la surface par un filin où se trouve attachée une bouée de couleur vive. Cela permet de repérer le casier de loin quand le moment est venu de le relever.

En programmation, c'est à peu près la même chose. Tant que vous ne vous éloignez pas du bord (entendez le pas 00), vous n'avez pas besoin de repère, mais si vous voulez réaliser des programmes longs et performants, vous aurez besoin de bouées pour retrouver le début des différentes phases.

L'étiquette : un point de repère, une bouée

Ces bouées prennent le nom d'étiquettes (labels en anglais). Vous allez les placer à chaque endroit du programme qui aura besoin d'être repéré pour l'exécution d'une phase particulière. Quand vous voudrez récupérer le contenu du casier (la portion de programme), il vous suffira d'aller à la bouée (l'étiquette) avec votre barque (le pointeur) et de tirer (lancer l'exécution).

Puisque nous sommes dans les affaires maritimes, nous allons réaliser un programme pour convertir les kilomètres en milles nautiques et inversement. Nous placerons une étiquette au début de chaque sous-ensemble pour pouvoir récupérer celui qui nous intéresse au moment voulu.

Traçons d'abord l'organigramme correspondant. Nous utiliserons comme symbole de l'étiquette le signe qui marque le début ou la fin d'un programme et nous inscrivons " Lbl " à l'intérieur pour éviter toute confusion :

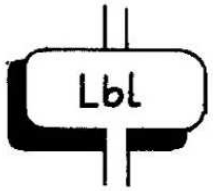


Fig. 1

Un identificateur de 0 à 9 numérotera les étiquettes. Sur la TI 57, l'identification est uniquement numérique. Sur d'autres machines, il peut être alphabétique, ou être un nom de fonction. D'autres matériels ne possèdent pas d'étiquettes. On doit dans ce cas se repérer sur les numéros de pas.

Voici un exemple de programme écrit pour TI 57 et utilisant deux étiquettes :

00	01	1
01	83	•
02	08	8
03	05	5
04	02	2
05	32 1	STO 1
06	86 1	2nd Lbl 1
07	45	÷
08	33 1	RCL 1
09	85	=
10	81	R/S
11	86 2	2nd Lbl 2
12	55	x
13	33 1	RCL 1
14	85	=
15	81	R/S

Pour exécuter le programme, faites RST puis R/S. Entrez une dis-

La programmation : les boucles et les tests

tance à convertir et faites GTO1, R/S si la conversion désirée est km en milles. Sinon faites GTO2, R/S après avoir rentré la valeur. Ce petit programme peut être utilisé pour convertir toute sorte de données, en remplaçant la constante du début de programme. Si par exemple, vous désirez convertir des centimètres en inches (mesure anglaise) et inversement, la constante est 2,54. Pour convertir des mètres en pieds, 0,305 ; des kilomètres en *miles* (attention, ce ne sont pas les mêmes que tout à l'heure, ici n'est la mesure anglaise) 1,609. Vous obtiendrez la conversion de francs français en dollars en utilisant 5,95. Mais attention, car ce n'est plus une constante, et il faut réajuster en fonction du cours du change...

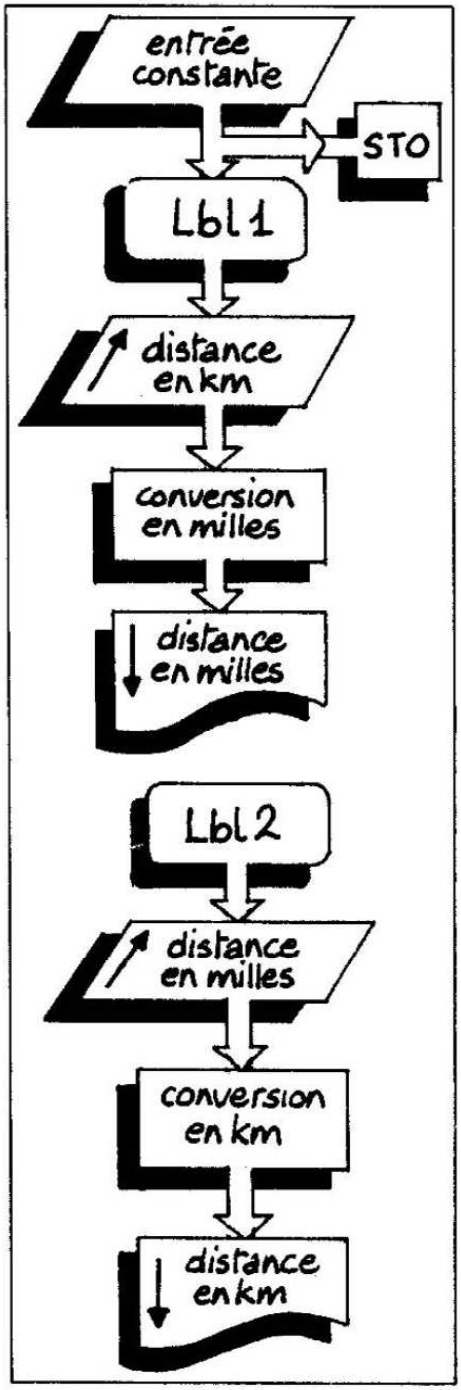


Fig. 2 - Organigramme de la conversion des kilomètres en milles et inversement.

—Un ordre impératif : Va !—

Nous avons lancé l'exécution des sous-ensembles du programme en utilisant la touche GTO suivie du numéro de l'étiquette que nous voulions appeler. GTO est l'abréviation de l'impératif anglais Goto qui signifie " va jusqu'à ". Cette instruction envoie le pointeur de programme sur l'étiquette dont le numéro est spécifié par l'identificateur de GTO.

Nous avons utilisé l'instruction en mode d'exécution, mais elle est également programmable. Placée à un endroit quelconque du programme, elle positionnera le pointeur à l'étiquette définie. C'est pour cela qu'on l'appelle *branchement obligé,* ou *inconditionnel.*

Cet ordre a deux utilisations principales. Il permet d'une part d'exécuter des bouclages lorsque le début de la boucle n'est pas au pas 00 (dans ce cas, nous avons déjà vu que c'est l'instruction RST qui est utilisée) et il est d'autre part employé pour sortir d'un test. Nous étudierons cette application un peu plus loin.

Si vous prenez l'habitude d'utiliser GTO en dehors de ces deux cas, c'est le signe que vous ne passez pas suffisamment de temps à étudier les algorithmes de vos programmes : ils manquent de rigueur.

Nous venons de voir que sur TI 57, l'ordre GTO renvoie nécessairement à une étiquette marquant le début d'une phase de programme. On parle dans ce cas-là d'adressage *relatif.* Ce mode est à opposer à l'adressage *absolu* qui existe sur beaucoup d'autres ordinateurs de poche (HP33, TI58-59, et dans tous les BASIC). L'identificateur de GTO est alors un numéro de pas, et le branchement se fait " à la volée ", sans qu'il n'y ait de repère spécial dans le programme.

L'adressage absolu présente l'avantage d'une plus grande rapidité de transfert, mais il complique en revanche beaucoup la mise au point des programmes puisque chaque insertion ou suppression de pas entraîne une modification des numéros de pas dans l'identificateur (ce n'est toutefois pas le cas en BASIC puisque les numéros de ligne ne changent pas lors de l'utilisation d'une commande d'édition).

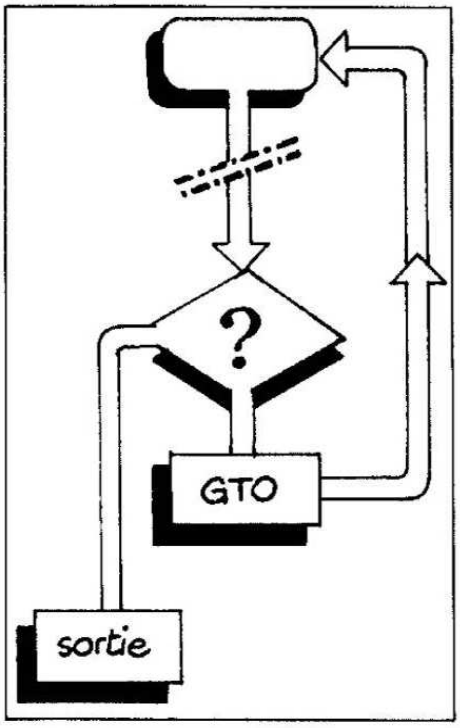


Fig. 3

Sur la TI 57, on peut cependant utiliser l'adressage absolu en mode calcul, pour envoyer le pointeur sur un pas donné avant de lancer l'exécution. Cette possibilité rend des services pour la mise au point des programmes. Il faut faire pour cela GTO 2nd nn, nn représentant les deux chiffres du numéro de pas.

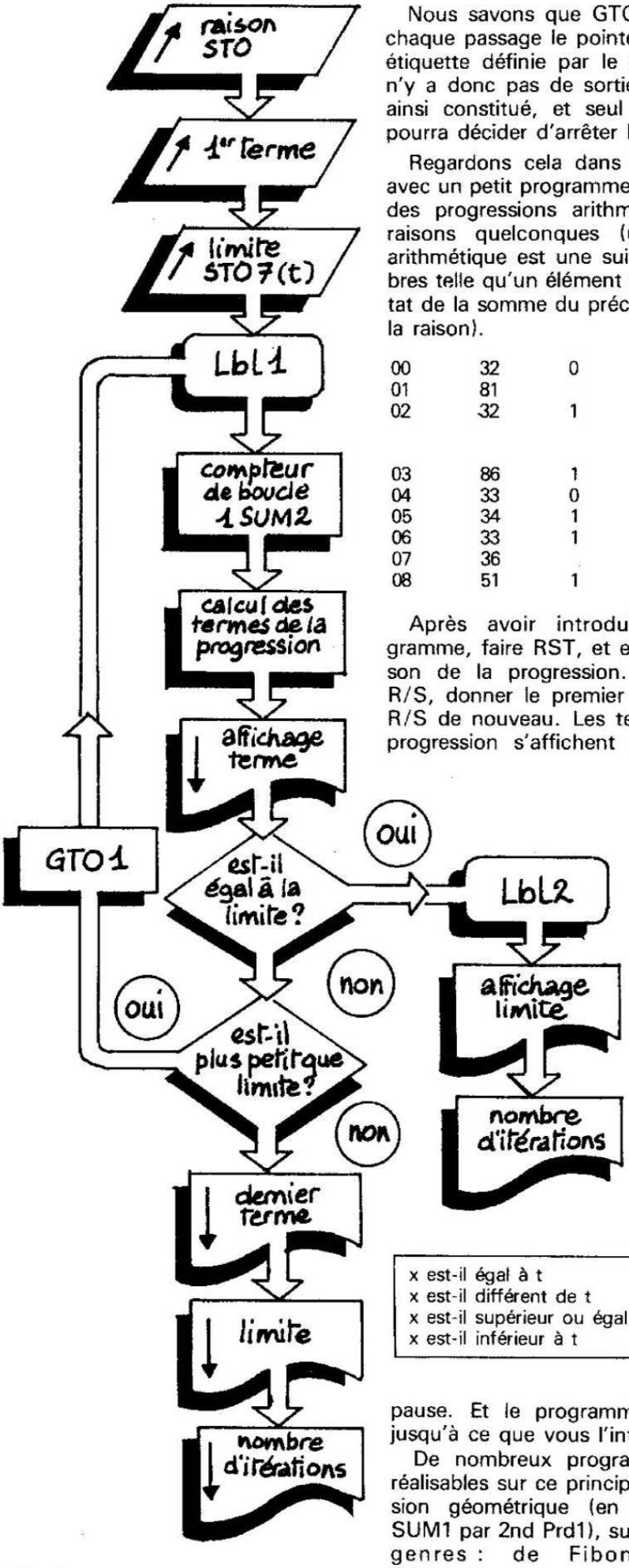


Fig. 4

Nous savons que GTO renvoie à chaque passage le pointeur sur une étiquette définie par le pointeur. Il n'y a donc pas de sortie au circuit ainsi constitué, et seul l'opérateur pourra décider d'arrêter la course.

Regardons cela dans la pratique avec un petit programme qui calcule des progressions arithmétiques de raisons quelconques (une raison arithmétique est une suite de nombres telle qu'un élément est le résultat de la somme du précédent et de la raison).

00	32	0	STO 0
01	81		R/S
02	32	1	STO 1
03	86	1	2nd Lbl 1
04	33	0	RCL 0
05	34	1	SUM 1
06	33	1	RCL 1
07	36		2nd Pause
08	51	1	GTO 1

Après avoir introduit le programme, faire RST, et entrer la raison de la progression. Après un R/S, donner le premier terme puis R/S de nouveau. Les termes de la progression s'affichent pendant la pause. Et le programme continue jusqu'à ce que vous l'interrompiez.

De nombreux programmes sont réalisables sur ce principe : progression géométrique (en remplaçant SUM1 par 2nd Prd1), suites de tous genres : de Fibonacci, de Riemann...

Nous pourrions même réétudier le " chronomètre " réalisé dans le précédent numéro à partir d'un algorithme similaire à celui que nous venons d'utiliser = mise en mémoire d'une constante, conversion décimale et addition en mémoire puis rappel et conversion sexagésimale avant l'affichage pendant une pause et enfin bouclage sur une étiquette située avant la conversion décimale.

Une sortie de secours

Tous ces programmes fonctionnent parfaitement, mais ils obligent à surveiller la calculatrice pendant tout son temps de travail jusqu'à ce qu'elle atteigne le résultat désiré. Et cela peut parfois prendre du temps. Fort heureusement, la calculatrice peut se surveiller elle-même. Et nous obtiendrons cela au moyen d'un test (voir fig. 3). Un losange a été ajouté à l'organigramme. A l'intérieur de ce losange, un point d'interrogation indique qu'une question est posée. Tant que la réponse est oui, le pointeur boucle par l'intermédiaire de GTO. Mais si la réponse change, le pointeur sort de la boucle et exécute une autre portion de programme. Pendant l'exécution d'un test, deux valeurs sont comparées et la question posée peut varier : la première donnée est-elle égale à la deuxième ? Est-elle différente ? Est-elle plus grande ou égale ? Plus petite ou égale ?

Dans la pratique, sur la TI 57 ces deux éléments de comparaison sont contenus l'un à l'affichage, l'autre dans une mémoire spéciale : le registre t. En fait, c'est la mémoire n°7 qui est utilisée dans ce cas-là. Et les tests possibles sont représentés dans le tableau ci-dessous.

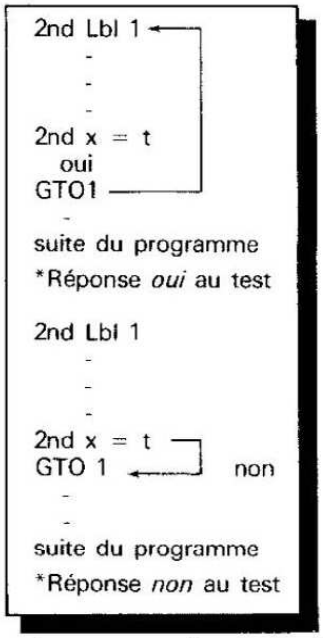
x est-il égal à t	$x = t$	noté sur la TI57	2nd x = t
x est-il différent de t	$x \neq t$	noté	INV 2nd x = t
x est-il supérieur ou égal à t	$x \geqslant t$	noté	2nd x ⩾ t
x est-il inférieur à t	$x < t$	noté	INV 2nd x ⩾ t

La réponse à la question posée peut être oui ou non. Si c'est " oui " le pas qui suit le test est exécuté. Il contient pratiquement toujours un branchement obligé qui assurera soit le bouclage du programme soit une autre opération. Si la réponse est négative, le pas sui-

La programmation : les boucles et les tests

vant le test est sauté et le pointeur suit son petit bonhomme de chemin comme si rien ne s'était passé.

Revoyons cela de façon un peu schématique :



Le registre t peut être chargé de différentes façons avant le test. L'accès normal est la boucle x ⇌ t qui échange le contenu de l'affichage et celui de t. Le résultat serait

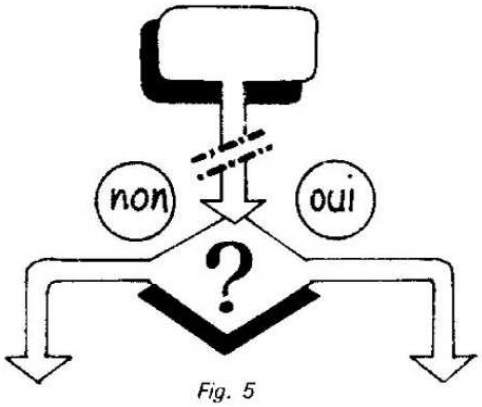


Fig. 5

le même en faisant 2nd Exc 7. Dans certains cas, il est utile de faire disparaître l'ancien contenu de t au lieu de le placer à l'affichage. Il faut alors simplement exécuter STO7.

Essayons maintenant de concrétiser tout cela avec un exemple de programme. Nous allons reprendre la progression arithmétique et rajouter un test pour obtenir un arrêt automatique d'exécution. Nous fixerons avant le départ une valeur limite qui provoquera la sortie de la boucle. Cette limite n'étant pas nécessairement un terme de la progression, nous exécuterons un test supplémentaire (voir fig. 4).

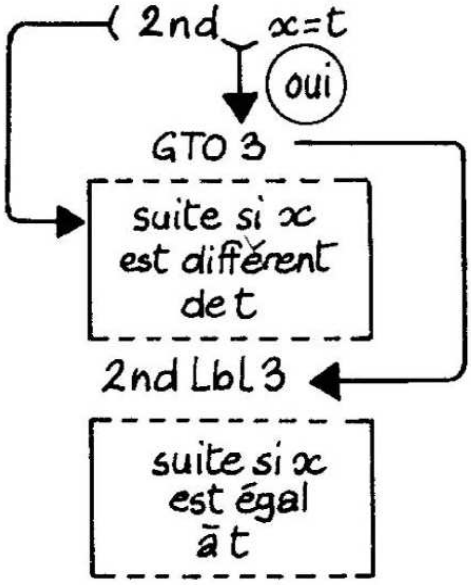


Fig. 6

Vous remarquez que le début du programme est semblable au précédent. Mais après l'affichage du terme de la progression qui vient d'être calculé, le premier test examine s'il est égal à la limite. Si c'est le cas, ce terme est affiché à nouveau, suivi par le nombre de tours de circuits parcourus. S'il n'y a pas égalité, un second test vérifie si le dernier terme obtenu est plus petit que la limite. Si la réponse est oui, un GTO réexpédie le pointeur au début de la boucle. Si la réponse est non, le dernier terme est affiché, puis la limite prévue de la progression et le nombre d'itérations effectuées. Ce deuxième test assure la sortie de la boucle en prévenant que la limite choisie ne fait pas partie de la progression.

Vous pouvez constater que le deuxième test contrôle si x est plus petit que t et non pas s'il est inférieur ou égal à t, comme cela est indiqué sur la touche : x ⩾ t. Ce résultat est obtenu par la succession des deux tests, le premier éliminant le cas de la stricte égalité, le second

00	32 0	STO 0	— Mise en mémoire de la raison
01	81	R/S	Arrêt, entrée du 1er terme de la progression
02	32 1	STO 1	Mise en mémoire
03	81	R/S	Arrêt. Chargement du registre de test avec
04	32 7	STO 7	la valeur limite de la progression
05	00	0	Remise à zéro du registre compteur
06	32 2	STO 2	de tour
07	86 1	2nd Lbl 1	
08	01	1	Comptage des itérations
09	34 2	SUM 2	
10	33 0	RCL 0	
11	34 1	SUM 1	Calcul du terme suivant de la progression
12	33 1	RCL 1	et affichage pendant
13	36	2nd Pause	une pause
14	66	2nd x = t	— Le terme est-il égal à la limite ?
15	51 2	GTO 2	— si oui aller à Lbl 2
16	-76	INV 2nd x ⩾ t	— si non est-il plus petit que la limite ?
17	51 1	GTO 1	— si oui boucler en allant en 1
18	33 1	RCL 1	— si non afficher le dernier terme calculé
19	36	2nd Pause	pendant une pause
20	22	x ⇌ t	Rappel du registre t :
21	36	2nd Pause	Afficher la limite choisie (qui dans ce cas ne fait pas partie de la progression)
22	33 2	RCL 2	nombre d'itérations effectuées
23	81	R/S	
24	86 2	2nd Lbl 2	
25	33 7	RCL 7	Afficher le dernier terme calculé
26	36	2nd Pause	
27	33 2	RCL 2	Afficher le nombre d'itérations
28	36	2nd Pause	Le 2nd Pause pourrait être supprimé, mais
29	81	R/S	il permet de différencier les deux circuits.

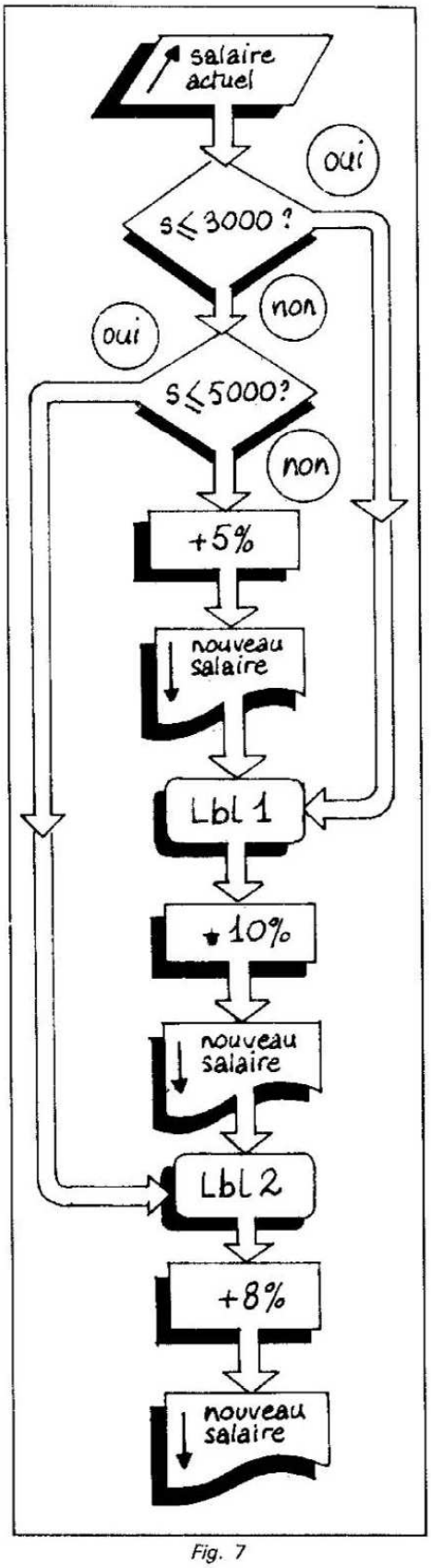


Fig. 7

00	22	x ⇌ t	Entrée du salaire actuel dans registre t
01	03	3	
02	00	0	3 000
03	00	0	
04	00	0	
05	76	2nd x ⩾ t	Est-ce plus grand que le salaire ?
06	51 1	GTO 1	— oui : aller à 1 ; sinon, on poursuit.
07	05	5	
08	00	0	
09	00	0	5 000
10	00	0	
11	76	2nd x ⩾ t	Est-ce plus grand que le salaire ?
12	51 2	GTO 2	— oui : aller en 2.
13	33 7	RCL 7	— non : on calcule donc le nouveau
14	75	+	salaire avec une augmentation de 5 %
15	14	CE	
16	55	x	
17	83	.	
18	00	0	
19	05	5	
20	85	=	
21	81	R/S	
22	86 1	2nd Lbl 1	Calcul du nouveau salaire dans le cas où
23	33 7	RCL 7	l'ancien était inférieur à 3 000
24	75	+	
25	14	CE	
26	55	x	
27	83	.	
28	01	1	
29	85	=	
30	81	R/S	
31	86 2	2nd Lbl 2	Calcul du nouveau salaire lorsque l'ancien
32	33 7	RCL 7	était compris entre 3 000 et 5 000 F.
33	75	+	
34	14	CE	
35	55	x	
36	83	.	
37	00	0	
38	08	8	
39	85	=	
40	81	R/S	

Charger le programme, faire RST. Entrer l'ancien salaire et R/S.

ne peut plus contrôler que la stricte inégalité.

Nous n'avons jusqu'ici considéré les tests que comme un moyen de sortir d'une boucle de programme. Vous avez sans doute constaté, dans le programme précédent, qu'ils peuvent servir à autre chose : proposer des voies différentes pour la suite du programme selon le résultat du test. Dans un cas une suite d'opérations sera effectuée, dans l'autre la succession sera différente, ce qui est schématisé à la figure 5.

Dans la pratique du programme cela se traduira très souvent par un GTO après le test envoyant le pointeur à une étiquette située plus bas dans le programme. Entre ce GTO et l'étiquette en question, l'autre voie de réponse au test est représentée par la figure 6.

Nous allons détaillé ce cas dans un exemple.

Un patron décide d'augmenter ses employés, en faisant varier le taux d'augmentation en fonction du salaire de chacun (fig. 7).

Il donne 10 % à ceux dont le salaire net est inférieur à 3 000 F, 8 % si le salaire est compris entre 3 000 et 5 000 F, et 5 % pour les rémunérations supérieures à 5 000 F. Le programme devra calculer le salaire après augmentation.

Vous rencontrerez dans ce programme une utilisation un peu particulière de la touche CE. Elle sert habituellement à effacer le registre d'affichage si il n'y a pas eu de signe opératoire appuyé. Ici, elle se comporte comme un rappel derrière le signe opératoire du contenu du registre d'affichage.

Essayez la séquence de touches 5,0,0, + , CE, = , le résultat est 1 000 ce qui montre que l'appui de CE a maintenu 500 à l'affichage pour l'opération.

Vous pouvez également constater que la même séquence de calcul est répétée 3 fois, avec relativement peu de différence (seul le taux d'augmentation varie). Cela représente un gâchis important de place en mémoire programme. Il est heureusement possible d'éviter cette réécriture en utilisant les sous-programmes. Et nous en parlerons une prochaine fois.

☐ Xavier de La Tullaye