Système conversationnel pour la documentation et l'utilisation de logiciels en hydrologie (1)

Thierry LEVIANDIER et Cécile LOUMAGNE (2)

RÉSUMÉ

On présente les problèmes rencontrés et les moyens développés pour rendre conversationnelle une bibliothèque étendue de programmes utilisés en hydrologie.

De nombreux programmes de cette bibliothèque n'ont pas été conçus pour être conversationnels et il serait trop coûteux de les reprendre entièrement. Par ailleurs, certains programmes ne peuvent être rendus entièrement conversationnels leur temps d'exécution étant trop long. On ne rend alors conversationnelle que la partie concernant l'entrée des options de calcul.

Pour cela on a établi un cahier des charges des besoins et nécessités conversationnelles et présenté une solution dérivée d'un programme gérant une documentation hiérarchisée de logiciels.

On peut représenter les options de calcul d'un programme par un arbre dont le parcours intégral constitue la documentation de ce programme et dont l'utilisation interactive consiste en un parcours guidé dans un menu et interrompu par la saisie de valeurs numériques.

Les fonctions interactives peuvent être créées en ajoutant simplement dans la documentation des commandes suspendant le parcours et le détournant vers des commandes de mise à jour de la base de données hiérarchiques.

On présente les problèmes rencontrés pour l'application de ce système aux logiciels d'un service de recherche en hydrologie. La structure arborescente s'est révélée suffisante pour traiter quelques cas concrets même si certaines améliorations pourront y être apportées notamment pour simplifier la tâche de l'utilisateur ainsi que pour la mise à jour de la base de données.

MOTS-CLÉS : Système conversationnel – Documentation hydrologique – Logiciels.

ABSTRACT

AN INTERACTIVE SYSTEM FOR DOCUMENTATION AND USE OF PROGRAMS IN HYDROLOGY

The paper describes tools developed for making interactive a large library of codes of hydrological interest and problems encountered. Many codes have not been thought in an interactive mind and it would be to expensive to rewrite them entirely. Besides, somes codes with long times of execution can be interactive only in the entry of parameters coding option of treatments. A list of interactive facilities necessary for such purpose is given and it is shown how it can be met by improvement of a program dealing with a hierarchical documentation of software.

The options of a program can often be represented by a tree, and interactive use of it consists in going through its menu-driven and answering questions, whereas going through all branches of the tree is equivalent to read the

(1) Ce texte est une version augmentée de la communication des auteurs : « An interactive system for documentation and use of programs in hydrology » présentée à : 2nd International Conference. Interactive Information and Processing Systems for Meteorology, Oceanography and Hydrology of American Meteorological Society, Janv. 1986. Miami, Floride (USA).

Pour plus de détails (mais portant sur une version légèrement remaniée depuis), voir le mémoire de Cécile Loumagne : « Logiciel général de saisie interactive d'options de calcul de programmes scientifiques », 1985 ; CEMAGREF – Université Paris-Sud Orsay.

(2) Division Hydrologie-Hydraulique, CEMAGREF, BP 121, 92164 ANTONY CEDEX.

directions for use. So interactive facilities can be implemented by adding in the text of directions for use commands meaning interruption of reading/writing and activation of (temporary) updating.

The problems induced in management of hierarchical data base are examined and it is investigated on practical case on which extent this framework is sufficient and which complementary tools are needed, specially when a greater initiative of user is required or for data management.

KEY WORDS : Interactive system – Documentation on water – Software.

1. INTRODUCTION

Les logiciels de calcul utilisés en hydrologie reposent sur deux types de données : d'une part, les données hydrologiques proprement dites, comme par exemple les chroniques de débit ou de pluies et des données constantes concernant tel bassin versant ou tel profil en travers d'une rivière, etc., et, d'autre part, les options de traitement qui orientent l'exécution du programme et indiquent les paramètres à saisir.

Il est inutile de revenir sur les avantages qu'offre en pareil cas le mode conversationnel. Toutefois, un programme conversationnel n'est pas aussi facile à écrire qu'il y paraît et, faute d'une réalisation très soignée, les utilisateurs préféreront rester fidèles aux anciens programmes de traitement par lots où les fichiers de données sont édités suivant les instructions énoncées dans une notice.

De plus, la réalisation d'un programme conversationnel est un travail long et coûteux et le programmateur, qui est avant tout hydrologue, n'y trouve pas d'intérêt majeur, encore moins s'il s'agit de convertir en mode conversationnel un programme de traitement par lots déjà ancien mais toujours opérationnel.

Le travail que nous présentons ici permet de créer, grâce à un programme interactif général, un module conversationnel de saisie de données qui peut être utilisé avant le programme de calcul proprement dit, lancé généralement en traitement par lots. Cette solution répond en outre au fait que les systèmes de la plupart des ordinateurs interdisent toute exploitation en temps partagé et toute interactivité aux programmes exigeant une grande capacité mémoire ou un temps de traitement trop long.

Ce module conversationnel permet donc de saisir des données quantitatives et des options de traitement, puis de les stocker dans un fichier avec le même format qu'auparavant et, pour les secondes, selon les mêmes conventions de codage. Les données en question sont souvent interdépendantes et plutôt logiques et symboliques que quantitatives.

2. CARACTÉRISTIQUES SOUHAITABLES D'UN MODULE CONVERSATIONNEL

En plus de capacités dont on ne peut faire l'économie et que nous ne détaillerons pas, comme par exemple le traitement adéquat de réponses incorrectes de la part de l'utilisateur, les caractéristiques essentielles que devrait avoir un programme conversationnel sont les suivantes :

- Le programme devrait à la fois fournir toutes les informations nécessaires au non initié et ne dire que l'essentiel à l'utilisateur expérimenté (il s'agit ici de connaissances sur le déroulement du programme lui-même et non pas de connaissances scientifiques).
- De manière plus générale, le programme conversationnel devrait servir aussi bien pour exécuter un programme de traitement par lots que pour seulement expliquer sa fonction.
- L'utilisateur devrait avoir la possibilité d'annuler toute réponse antérieure par un simple retour en arrière.
- L'utilisateur devrait être en mesure d'exécuter un programme plusieurs fois de suite, avec quelques variantes, sans avoir à répondre systématiquement à toutes les questions.
- Le texte du menu conversationnel devrait pouvoir être aisément remis à jour, soit lorsque le programme de traitements par lots l'exige, soit à la suite de suggestions et de réclamations de la part des utilisateurs.
- Il devrait être possible de prévoir des versions simplifiées pour certains utilisateurs qui n'ont pas besoin de toutes les options de traitement.

Il est clair que la « quantité » de logiciel alors nécessaire dépasse largement celle qui suffirait à une application spécialisée de complexité moyenne ne satisfaisant que les exigences élémentaires du mode conversationnel. La mise à jour des données étant plus simple que la mise à jour des programmes, à plus forte raison des programmes compilés,

nous avons donc été amenés à écrire un programme très général générant des modules conversationnels dont les données propres à une application spécifique sont stockées dans un fichier arborescent.

La figure 1 schématise les modifications apportées (partie de droite) à la situation initiale se présentant à l'utilisateur et au programmeur (partie de gauche) :

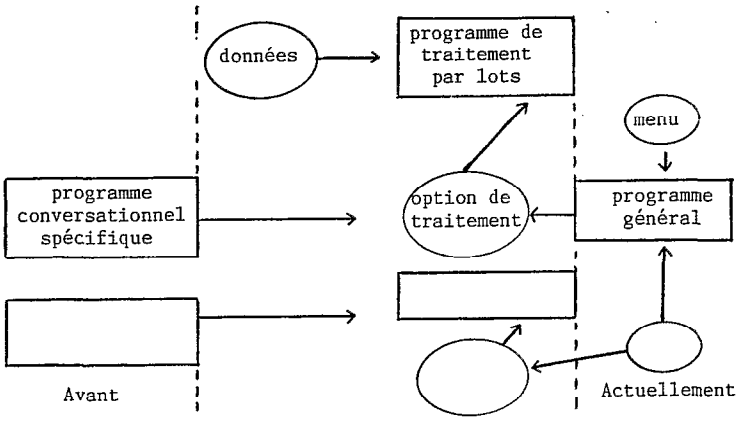

FIG. 1

Dans un tel schéma, on ne sait plus très bien si le programme général est un logiciel ou un « compilateur » et si les données du menu sont vraiment des données ou plutôt des programmes. Cette ambiguïté n'est pas rare en informatique mais les utilisateurs n'y sont pas toujours habitués.

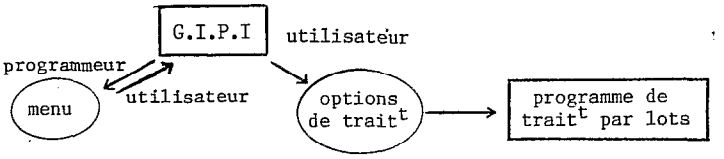

FIG. 2

Le programme général permet d'éviter la multiplication de programmes conversationnels que les programmateurs doivent utiliser pour générer leurs données. Ce programme général, nous l'appellerons désormais : « générateur interactif de « programmes » interactifs » ou « G.I.P.I. ».

Il est vrai que le G.I.P.I. ne remplit que partiellement les conditions requises. Cependant, il ouvre la voie à des améliorations futures. Il a été conçu à partir d'un programme de consultation d'une documentation hiérarchisée que nous allons brièvement décrire en nous attachant aux points qui nous intéressent directement.

3. PROGRAMME DE CONSULTATION D'UNE DOCUMENTATION HIÉRARCHISÉE

Si la documentation hiérarchisée offre l'avantage de la simplicité, elle n'offre pas celui de la rapidité. D'autre part, elle repose sur une structure arborescente arbitraire plus parlante pour ceux qui la conçoivent que pour ceux qui doivent l'utiliser.

Nous utilisons ce principe de documentation depuis quelques années pour la documentation de logiciels d'hydrologie qui se trouvent dans notre centre (fig. 3), et dénommé HYSVP.

L'ensemble des informations est divisé en sous-ensembles, eux-mêmes divisés en sous-ensembles... Toutes les branches de l'arbre ne sont en fait que des chapitres et des sous-chapitres, et seules les feuilles sont porteuses d'information directement utilisable. L'utilisateur peut, soit parcourir l'arbre pas à pas, soit le parcourir en entier. Lorsqu'il n'est pas sûr de son parcours, il peut, sans perdre de temps, effectuer un balayage de l'ensemble des branches sans leurs feuilles. Il peut également remonter vers la racine en une seule opération.

1. COMMENT UTILISER CETTE DOCUMENTATION
2. DOC. EXTERIEURE OU NON AUTOMATISEE
    1 CII, CTI ET DIVISION CALCUL
    2 HYDROLOGIE, PRESENTATION GENERALE
    3 HYDROLOGIE, NOTICES DE PROGRAMMES
    4 HYDRAULIQUE FLUVIALE, NOTICES DE PROGRAMMES
    5 HYDRAULIQUE SOUTERRAINE, NOTICES DE PROGRAMMES
    6 HUMIDIMETRIE, NOTICES DE PROGRAMMES
    7 HYDROLOGIE, AFFICHAGE MURAL
    8 RECOMMANDATIONS AUX PROGRAMMEURS
3. CLASSEMENT THEMATIQUE DES PROG. DISPONIBLES
    1 PARCOURS DE RESEAU HYDRO. ET EXTRAC. CODES
    2 MANIPULATIONS DE DONNEES HYDROLOGIQUES
        1 TRANSCODAGE ET CHANG. PAS DE TEMPS FIXE
        2 DECOUPAGE DE PAS DE TEMPS VARIABLE
        3 EDITION D'ANNUAIRES DE DONNEES JOURNALIERES
        4 EXTRACTION DE VAL. CARACTERIST. DE DEBITS
        5 EXTRACTION DE PLUIES SUP. SEUIL A P.D.T. VARIABL
        6 EXTR. DE VALEURS EXTREM. ET MAXI SUR DON. JOURN
        7 DETERMINATION D'EPISODES SECS ET PLUVIEUX
        8 CALCUL DE FLUX
    3 STATISTIQUES ET MODELES STOCHASTIQUES
        1 AJUSTEMENT DE FREQUENCE
            1 CAS GENERAL
            2 EXTREMES DE PLUS. JOURS A PARTIR FICH. JOURN
        2 AJUSTEMENT INTENSITE DUREE FREQUENCE
        3 CORRELATIONS ET COMPARAISON DE 2 CHRONIQUES
            1 FONCTIONS DISPONIBLES
            2 PROCEDURES A UTILISER
        4 REGRESSION MULTIPLE
            1 REGRESSION LINEAIRE
            2 REGRESSION PAR BOULES
        5 ANALYSE DE PROCESSUS
        6 ANALYSE DE DONNEES
    4 GRAPHIQUES
        1 TRACE D'UNE COURBE EN FONCTION DU TEMPS
        2 GRAPHES D'ECARTS CUMULES
        3 TRACE D'ISOVALEURS
        4 TRACE DE POLYGONES (ET PRISMES) DE THIESSEN
        5 VUE PERSPECTIVE DE SURFACE
    5 MODELES HYDROLOGIQUES DETERMINISTES
    6 GESTION DE RESERVOIRS
    7 HYDROELECTRICITE
    8 HYDRAULIQUE FLUVIALE EN PERMANENT
        1 GEOMETRIE SIMPLIFIEE ET UN SEUL BIEF
        2 GEOMETRIE REELLE ET MAILLAGE
            1 NOTICE D'UTILISATION : PO54NOTICE / TS18
            2 PROCEDURES / TS18 DE MISE EN OEUVRE

FIG. 3. – Liste partielle du menu présenté en chapitres et sous-chapitres (HYSVP) (extrait des Informations Techniques n° 3, cahier 54, juin 84).

Les principales commandes, selon une terminologie généalogique, sont les suivantes :

- i : (nombre entier positif) pour choisir la nième ligne du menu ou le nième fils du nœud courant.
- — 1 : retour au menu père.
- + et — : pour aller respectivement vers le frère aîné et le frère cadet.
- > : pour tous les descendants.

4. MISE A JOUR DE LA DOCUMENTATION

Il est nécessaire de permettre l'insertion de nouvelles branches et de nouvelles feuilles au fur et à mesure que le volume d'information augmente et surtout de prévoir la possibilité de modifier la structure arborescente. Voici par exemple deux commandes qui donnent cette possibilité :

- Ci : crée i (i entier $\geqslant 0$). Création d'un nouveau fils entre le nième et le nième + 1 fils du nœud courant ; le programme demande le texte qui doit être saisi ; le nœud courant est inchangé.
- Vi : suivi de jG, ou bien iV suivi de Gj, sont utilisés pour déplacer un rameau (et tous ses descendants) d'un nœud à l'autre.
  avec : i, j : fils du nœud courant, peuvent être égaux à zéro si le nœud courant n'a pas de fils.
- V : viser, doit précéder.
- G : greffer ; si le nœud est à gauche de la commande, il est alors l'origine (là où l'arbre doit être coupé) ; s'il est à droite, il est alors la cible.

Cette dernière commande est double puisque nous ne nous référons qu'aux nœuds reliés au nœud courant sans jamais utiliser leur adresse absolue. Il suffit à l'utilisateur de parcourir l'arbre entre l'exécution de ces deux commandes, de l'origine vers la cible, ou vice versa.

5. ANALOGIE ENTRE UN PROGRAMME CONVERSATIONNEL ET UN PROGRAMME DE CONSULTATION ET MISE A JOUR D'UNE DOCUMENTATION HIÉRARCHISÉE

L'utilisation du programme conversationnel ressemble fort au parcours de la structure arborescente d'une documentation, à deux exceptions près : arrêt sur un menu avec entrée sur clavier ou saisie de données (cf. tabl.).

Programme conversationnel	Consultation et mise à jour d'une documentation
. un menu est proposé si l'ascendant a été choisi dans un menu de rang supérieur	.idem
. programme à parcours automatique	.parcours automatique seulement sur demande de l'utilisateur
. entrée d'une ou de plusieurs valeurs numériques	.mise à jour de la documentation

Le programme conversationnel doit aussi être capable de mémoriser le parcours et de le transcrire sous une forme susceptible d'être lue par le programme de traitement par lots. En fait, cette capacité de mémorisation existait déjà dans le programme de consultation et mise à jour, et permettait de réorganiser la représentation interne de l'arbre.

Le programme de consultation d'une documentation hiérarchisée peut aussi être aisément transformé en un générateur interactif de programmes interactifs (G.I.P.I.). Il suffit d'y ajouter :

- Un caractère spécial (dans les données) capable d'interrompre le parcours et d'interpréter le texte, soit comme une commande de consultation, soit comme une commande de mise à jour, ou bien, s'il n'y a pas de texte, de permettre à l'utilisateur de reprendre la main.
- Une commande pour permettre aux utilisateurs de lancer le programme conversationnel et aux programmeurs de lancer le mode « consultation/mise à jour » pour constituer des fichiers spécifiques à des applications précises.
- Le format de sortie.

6. CAS ÉLÉMENTAIRES

Lors du parcours d'un arbre binaire pas à pas (fils aîné ; s'il n'y en a pas, frère cadet ; s'il n'y en a pas, père), il faut passer deux fois par des nœuds non terminaux et toute opération (impression du menu par exemple) peut être exécutée, soit au premier, soit au deuxième passage. Dans un programme de consultation d'une documentation hiérarchisée, c'est lors du premier passage que toute opération est exécutée. Pour le G.I.P.I., nous avons dû définir trois commandes d'interruption de parcours : une au premier passage (aller), une au deuxième passage (retour) et une au premier et/ou deuxième passage (toujours).

Dans notre programme, nous avons utilisé des crochets et des points virgules, mais pour plus de clarté, nous n'utiliserons pas de symboles dans cet exposé. Dans certains cas, notamment aux nœuds terminaux, on peut utiliser deux de ces commandes indifféremment. Dans un premier temps, nous allons expliquer cette méthode pour les cas élémentaires d'un programme conversationnel.

6.1. Acquisition des données (fig. 4)

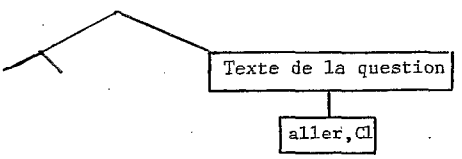

Fig. 4

Après l'affichage du texte de la question, « là » interrompt le parcours et interprète la commande « Cl ». La réponse à la question est entrée au clavier comme étant un second fils du nœud courant « texte ». Puis le parcours reprend.

6.2. Choix multiples (fig. 5)

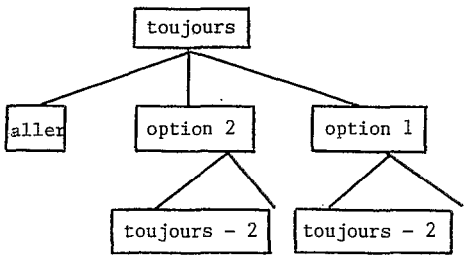

Fig. 5

- « toujours » permet l'impression de toutes les options et non pas uniquement l'option 1 et ses descendants.
- « aller » interrompt le parcours jusqu'à ce qu'une donnée soit saisie au clavier.
- « toujours - 2 » permet un retour en arrière sur le premier « toujours » et le choix d'une autre option.

Lorsque l'utilisateur ne désire plus choisir une autre option, il entre – 1 au clavier.

6.3. Choix exclusif

Comme précédemment, mais en remplaçant « toujours » par « aller ».

6.4. Boucle (fig. 6)

Les commandes d'arrêt de parcours appliquées aux commandes « V » et « G » produisent des boucles représentées comme suit :

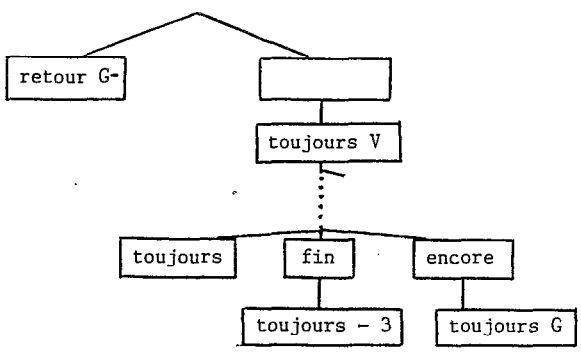

Fig. 6

Lorsqu'on arrive à « toujours G » la structure de l'arbre devient incohérente, en effet, le père du nœud courant devient son fils ! Celui-ci devient alors le nouveau nœud courant. Il a donc fallu modifier le programme de consultation d'une documentation puisqu'en temps normal cette opération est absolument interdite afin d'éviter des boucles inopportunes dues à une mauvaise mise à jour. La commande « retour G- », à la sortie de la boucle, rend de nouveau la structure arborescente cohérente en ne détruisant qu'une seule flèche à l'intérieur de la boucle.

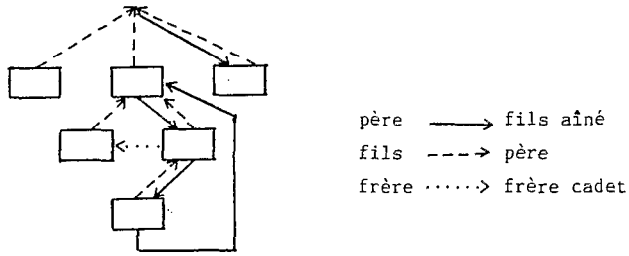

FIG. 7. – Structure arborescente temporairement incorrecte.

6.5. ANNULATION DES RÉPONSES INCORRECTES

L'intégralité du parcours est mise en mémoire, ce qui permet de refaire le parcours en sens inverse (étape par étape). L'annulation de données incorrectes acquises dans une boucle au cours du parcours n'est pas tout à fait opérationnelle actuellement.

6.6. UTILISATION MULTIPLE DE SOUS-ARBRES (SOUS-PROGRAMMES)

Il arrive fréquemment que la structure hiérarchique ne puisse être obtenue qu'au prix d'une répétition.

Supposons par exemple qu'un programme offre plusieurs options graphiques et, qu'au cours d'une exécution, une au plus puisse être utilisée. Dans le cas où un traitement graphique est effectivement demandé, l'utilisateur doit entrer quelques paramètres complémentaires (dimensions du dessin par exemple). On peut conserver la structure hiérarchique en insérant d'abord la question : « demanderez-vous un dessin ? » mais cela risque de dérouter l'utilisateur s'il ne sait pas encore quels dessins il est susceptible de commander. Il est donc préférable que la séquence interactive commune à tous les traitements graphiques apparaisse bien en sous-arbre de chacune des options graphiques, mais la répétition pure et simple nuit à la facilité de mise à jour ultérieure et prend de la place. Deux nouveaux symboles spéciaux ont donc été réservés pour définir et rappeler, d'un autre point, des sous-arbres identifiés par une chaîne de caractères. A la sortie du sous-arbre ou sous-programme, on retourne évidemment au point d'appel.

Une des premières applications est de constituer en sous-programme le sous-arbre codant la fin d'une boucle contrôlée par l'utilisateur.

7. ASSOCIATION DE PLUSIEURS CAS ÉLÉMENTAIRES

Le choix, exclusif ou non, d'une option de traitement ne modifie pas la structure arborescente et peut se faire à n'importe quel moment. L'acquisition des données peut aussi se faire à l'intérieur d'une boucle (toutes les valeurs acquises deviennent des frères dans l'arbre). Apparemment, les boucles peuvent s'emboîter les unes dans les autres sans grande difficulté (gestion d'une pile) mais un tel cas de figure ne s'est pas encore présenté étant donné que les applications ne requièrent pas, pour le moment, l'acquisition en série de grandes quantités de données.

8. FORMAT DU FICHIER DE SORTIE

Il faut incorporer dans l'arbre, à la suite du texte du menu, des indicateurs pour coder le fichier de sortie conformément au format prévu par le programme de traitement par lots.

Si la sortie a la longueur d'un enregistrement, il suffit de préciser, d'une part, dans quelles colonnes les valeurs saisies doivent figurer pour être correctement interprétées par le programme de traitements par lots et, d'autre part, quels symboles seront utilisés pour indiquer à ce programme les choix des options de traitement. Pour la saisie des données, aucun symbole n'est vraiment nécessaire, même si nous avons adopté le « ? ».

Ces éléments d'information sont invisibles à l'utilisateur ; il n'apparaît sur l'écran que le texte du menu (fig. 8).

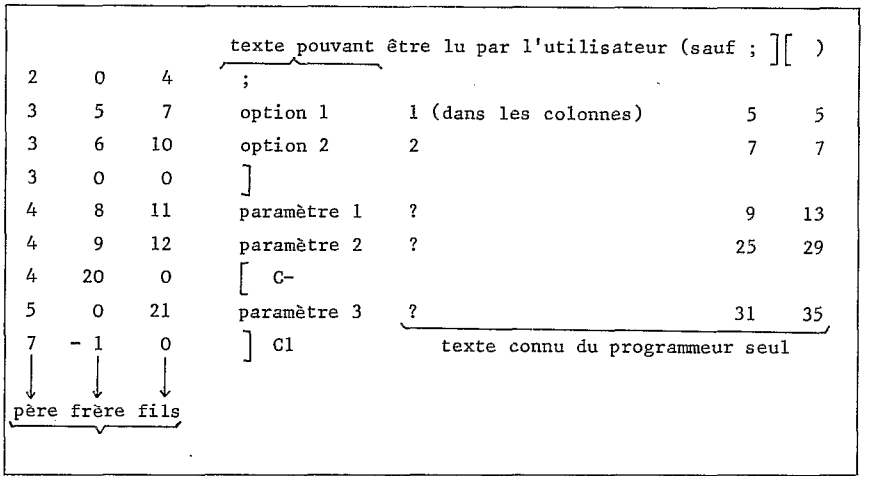

FIG. 8. – Représentation interne de l'arbre, connue seulement du programme.

8.1. PERFECTIONNEMENTS A VENIR

Actuellement, le programmeur peut utiliser les commandes de mise à jour pour constituer ce fichier arborescent, y compris le côté droit de la figure 8, mais il n'a aucune aide logicielle pour placer les chiffres au bon endroit, inconvénient qui peut être atténué si le terminal dispose d'un compteur de colonnes. Le G.I.P.I. sera vraiment conversationnel lorsque le programmeur n'aura plus à connaître ni ce format, ni la configuration des boucles et des menus (exclusifs ou non). Cela incombera à des sous-programmes ayant eux aussi une structure arborescente et possédant en outre une quatrième commande d'arrêt de parcours automatique. Cette commande modifiera le sens de la commande suivante en l'appliquant au nœud courant du sous-programme appelant le sous-programme courant. Par ailleurs, un tel sous-programme sera capable, grâce à une nouvelle commande, de recopier des instructions de lui-même dans le sous-programme appelant. Par exemple, le programmeur n'aura qu'à appeler les sous-programmes « boucle » ou « choix non exclusif », après s'être placé sur le nœud approprié.

9. APPLICATION A DES CAS CONCRETS

Nous avons constitué les fichiers correspondant à deux programmes différents. Le premier est un programme conversationnel, le second utilise les instructions d'une notice. La figure 9 présente un exemple d'application avec :
– extrait du fichier initial connu seulement par le programmeur ;
– extrait de l'exécution vue par l'utilisateur et résultat de l'exécution.

L'association des différents cas élémentaires, décrits précédemment, s'est révélée suffisante pour leur donner une structure arborescente. Toutefois, cela a posé quelques problèmes pour le programme conversationnel : la séquence selon laquelle apparaissent les questions a dû être modifiée pour remédier à leur dépendance fortuite dans le programme conversationnel initial qui a ainsi été révélée. Le nouveau programme conversationnel est plus explicatif que l'ancien, d'une part, parce que l'interdépendance a disparu et, d'autre part, parce que les questions oui/non sont devenues des menus de deux lignes (le rétablissement des questions oui/non est envisagé).

10. CONCLUSION

Une structure arborescente ne peut certes pas s'appliquer à toute saisie de données ou à tout choix d'options de traitement, ou du moins, il est vain d'espérer obtenir par cette méthode des programmes toujours compréhensibles. Mais notre intention n'a jamais été de concevoir un programme qui réponde à tous les problèmes qui peuvent

se poser. En effet, l'utilisation d'un tel programme est déconseillée pour les cas trop compliqués qui présentent par ailleurs une interdépendance des options de traitement, interdépendance liée plus à leur mise au point et à leur enrichissement progressif qu'à des raisons intrinsèques.

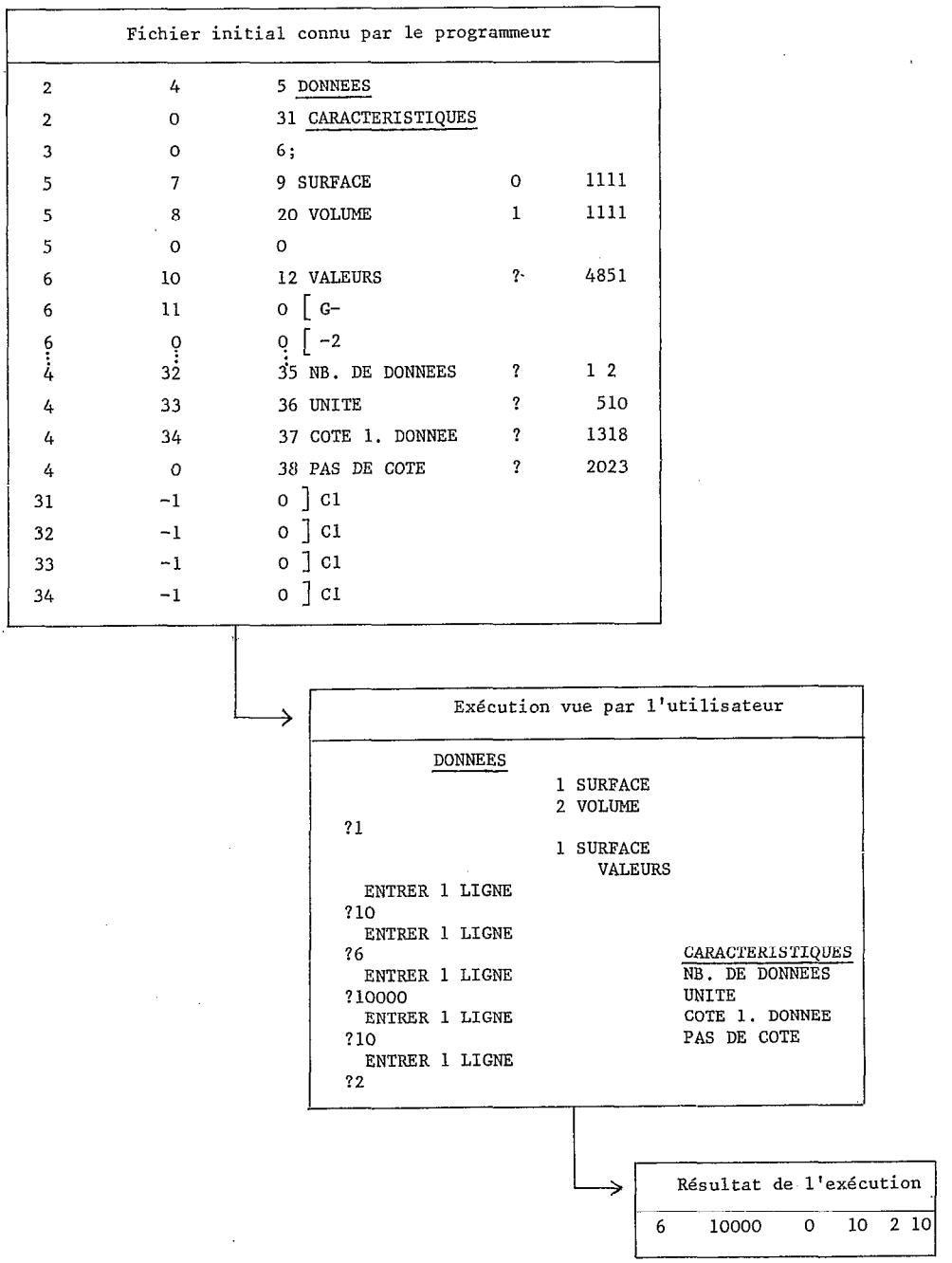

FIG. 9. – Extrait du mémoire de C. Loumagne, référence n° 2.

Malgré cela, les exemples traités sont de bon augure pour une extension à la majorité des programmes du Service. Les caractéristiques présentées précédemment comme souhaitables ne sont pas totalement acquises. Cependant, une représentation homogène a été trouvée pour tout le matériau interactif (questions, réponses possibles,

sorties correspondantes) et des opérations plus complexes peuvent être programmées sur ce matériau contenu dans des fichiers selon le format de la figure 8.

La structure hiérarchisée peut sembler contraignante et arbitraire. Son intérêt essentiel ici est qu'il est facile de définir sur cette structure un algorithme de parcours nécessaire dans des applications où l'initiative doit être beaucoup plus du côté du programme que du côté de l'utilisateur. Il serait en effet inutile et frustrant de laisser l'utilisateur demander des traitements qui ne sont pas programmés. Lorsque, dans un domaine bien cerné, l'utilisateur peut formuler sans risques toutes les questions, il faut au contraire viser un langage le plus souple possible, s'approchant du langage naturel. Ce deuxième point de vue semble pour l'instant réservé à la gestion, ou même seulement à l'interrogation des bases de données ; mais à plus long terme, il prévaudra certainement, ou bien une fusion des deux points de vue s'opérera.

Manuscrit accepté par le Comité de Rédaction le 8-8-1986