AVRIL 1982 N° 6

l'ADELIEN

BULLETIN de l'ADELI

ASSOCIATION LOI 1901 - 33, AVENUE DES GOBELINS, 75013 PARIS - TÉL. : 336 49-19

TABLE DES MATIERES

PRESENTATION DE L'ASSEMBLEE GENERALE.

NOTRE REUNION DE PRINTEMPS DE LA LOGIQUE INFORMATIQUE.

Au retour des vacances pascales et avant les préparatifs des grands départs d l'été , nous vous invitons à célébrer le renouveau de la Logique Informatique lors de notre Assemblée Générale.

Cette Assemblée se tiendra le mercredi 21 Avril 1982 à 14 heures à l'Hote SHERATON - 19,rue du Commandant René MOUCHOTTE , PARIS XIV éme.

Au programme , les thèmes traditionnels et obligeatoires des Associations régie par la loi de 1901

- Rapport moral du Président ,
- Rapport financier du Trésorier ,
- Renouvellement des membres du Comité.

Les candidats souhaitant s'informer sur leur futur role au sein du Comité son invités à prendre contact avec le Bureau de l'ADELI.

Ensuite , nous explorerons ensemble la voie de "la pierre philosophal informatique".
Depuis plusieurs lustres , tous les "alchiminformaticiens" mijotent sur leur fourneaux des recettes perpétuellement remaniées , afin de découvrir les mystére de la transmutation informatique.
Comment peut-on à partir des besoins confus et informels des Utilisateurs générer un systéme informatique (rapidement et au moindre cout) qui les satisfass pleinement ?

Sans prétendre à cette panacée , des outils , certes moins ambitieux mais plu réalistes , proposent une aide à la programmation.
Parmi ces outils , certains s'appuient sur la Logique Informatique. Nous e présenterons quelques'uns , réalisés par des membres de l'ADELI, au cours de l Table Ronde qui suivra l'Assemblée.
Cette réunion s'efforcera de mettre en évidence les principes de conception e appréciera les avantages des solutions préconisées , quant aux qualités attendue d'un logiciel d'application : Utilité , Facilité d'emploi , Fiabilité , Souplesse , Economie de réalisation et d'exploitation.

La réunion est ouverte à tous les adhérents de l'ADELI et à leurs invités , dan la mesure des places disponibles.

LES TRAVAUX DE LA DIVISION INFORMATIQUE DE LA CII-HB.

(texte communiqué par P.H. PETIT)

J.D. WARNIER a quitté la compagnie CII-HB à la fin de l'année 1981 et s consacre maintenant entièrement à ses travaux.

Il corrige les ouvrages qu'il a publiés et en prepare de nouveaux. Il répond au invitations qu'il reçoit de différents pays.

Au 1er Janvier 1982 , Pierre-Henri PETIT a pris la responsabilité de la Divisio Logique Informatique. L'équipe continue , en relation étroite avec J.D. WARNIE les travaux d'expérimentation et de diffusion et assure la publication réguliér de bulletin Etudes et Réalisations qui sert de lien aux différents praticiens o enseignants de la logique informatique. Ce bulletin peut etre demandé a secrétariat de la division :

CII-HONEYWELL BULL
Melle Isabelle PEPOUEY
Division Logique Informatique
94, avenue Gambetta
P.C. 1 U 078D
75020 PARIS

Tel : 360 02 22 Poste 22.11

LA STRUCTURE REPETITIVE TRADITIONNELLE EST-ELLE SUFFISANTE ?

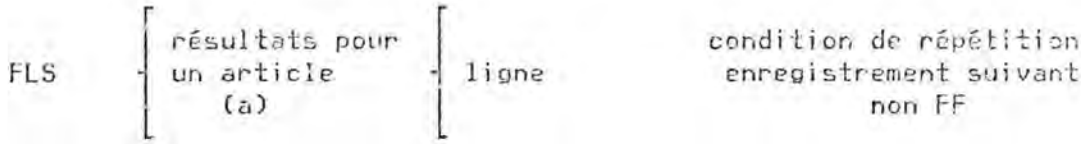
LA STRUCTURE REPETITIVE TRADITIONNELLE.

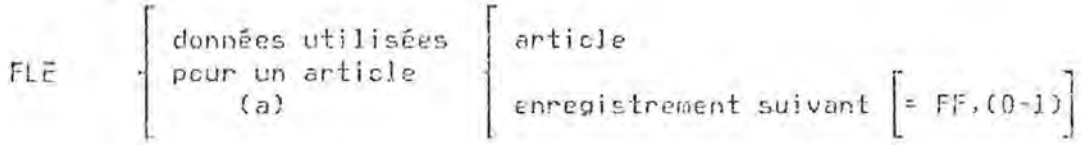
La structure répétitive traditionelle LCP (analogue à la structure DO UNTIL) traite un ensemble ordonné, composé de n (n > 0) éléments identiques.

En particulier, le traitement du dernier élément de l'ensemble appelle l'utilisation de données de même structure que les éléments qui le précédent.

Examinons le problème simple de l'édition d'un fichier séquentiel constitué de a articles (a > 0) et d'un enregistrement "fin de fichier" Il est clair que l'enregistrement "fin de fichier" n'appartient pas à l'ensemble des articles.

La construction du programme se déduit de la démarche suivante :





La séquence 20 traite le i ème article, acquiert l'enregistrement suivant de rang i+1, qui est alors identifié :

- soit comme un article,

- soit comme une marque de fin de fichier.

La détection de la fin de fichier entraine l'arrêt de la répétition.
L'identification d'un nouvel article entraine une nouvelle exécution de la séquence 20.

Cette structure répétitive traditionnelle, largement satisfaisante pour résoudre les problèmes liés à des ensembles composés d'éléments identiques (c'est à dire faisant appel à des données de même structure) se révèle souvent inadaptée pour traiter le cas le plus général de répétitive.

STRUCTURE REPETITIVE LA PLUS GENERALE.

Considérons maintenant l'ensemble ordonné le plus général constitué de n éléments (n > 1) dont le dernier utilise des données différentes des n-1 précédents.

La condition de détection de ce dernier élément utilisé est évidemment identique à celle de détermination de la fin de la répétitive.

La résolution de ce cas général est toujours possible à l'aide de structures alternatives et répétitives classiques, en introduisant des séquences parasites (logiquement vides).

Ces séquences, parfaitement inutiles, alourdissent toujours la programmation en place occupée et souvent en temps d'exploitation.
Mais, il existe une technique d'emploi certes plus délicat, qui allie les avantages de la démarche logique informatique à des performances optimales.

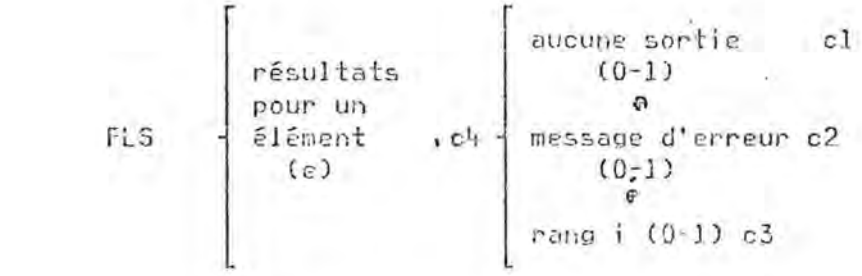
EXEMPLE BIEN CONNU

Considérons la recherche séquentielle d'une valeur V dans une table T de n éléments différents, non classés.
On compare successivement V à tous les éléments (en commençant par celui de rang 1) de la table.

Pour chaque élément, il y a 3 éventualités de sortie :

- aucune sortie si Ti non égal V et i < N
- affichage d'un message d'erreur si Ti non égal V et i = N
- affichage du rang i si Ti = V

Ce qui peut s'exprimer sur le FLS :



c1 , c2 , c3 et c4 sont les conditions de sortie ou de répétition

c1 : Ti non = V et i < N

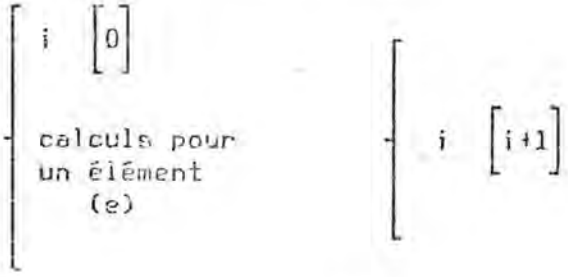
c2 : Ti non = V et i = N

c3 : Ti = V

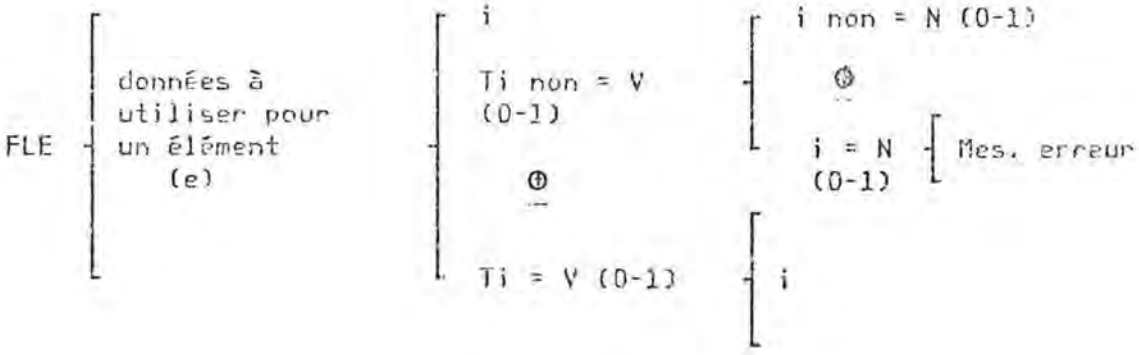
c4 : Ti non = V et i < N

c4 est identique à c1

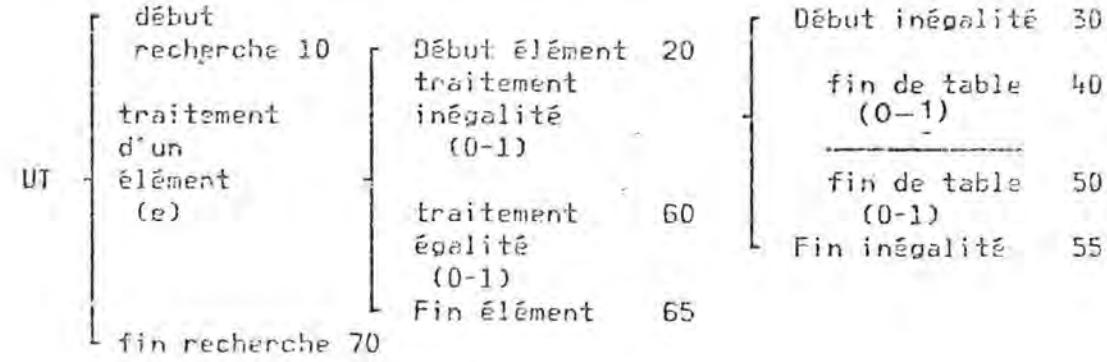
La donnée interne i est générée par le traitement



Les données à utiliser s'organisent selon :



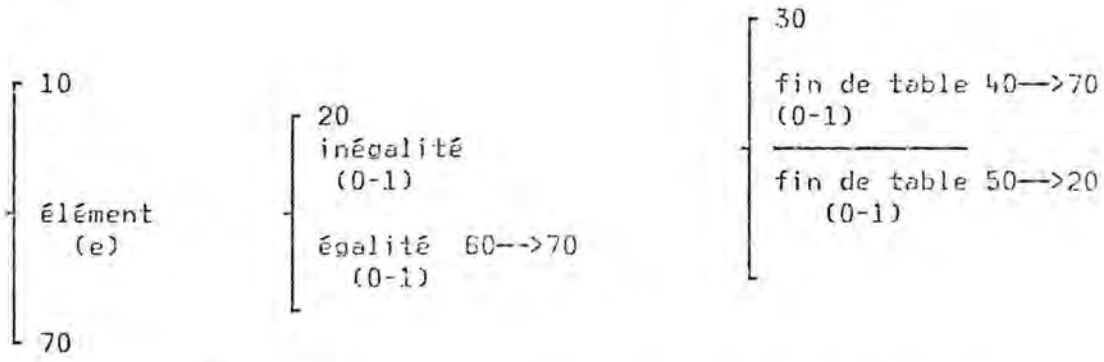
Ce qui conduit à la structure du programme :



La condition de répétition en 65 est identique à celle de l'exécution de séquence 40.

La condition d'arrêt de la répétition est la réunion des conditio d'exécution de séquence 50 et 60.

Logiquement les séquences 55 et 65 seront toujours vides, bien qu'elles soient suivies dans le formaliste habituel par des branchements inconditionnels relatifs à des conditions déjà exprimées. La structure du programme peut alors s'écrire :

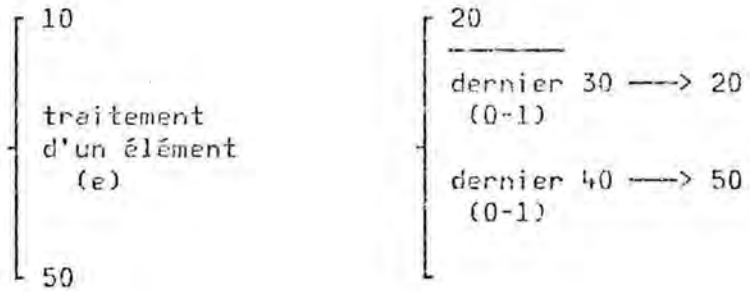


50 —>20 implique un branchement inconditionnel de 50 vers 20.

EXPRESSION DE LA STRUCTURE REPETITIVE.

On peut toujours représenter les données à utiliser pour un traitement répétitif par :

qui conduit à la structure de programme :



EXERCICE PROPOSE.

Sélection d'articles dans un fichier

Les articles d'un fichier contiennent, en particulier, n zones Z de même format. Pour chaque zone, une liste de valeurs souhaitables est proposée.

Un article est sélectionné si chacune de ses zones Z contient une valeur souhaitable. Un article n'est pas sélectionné si l'une de ses zones Z contient une valeur n'appartenant pas à la liste de ses valeurs souhaitables.

Comparer les solutions obtenues à l'aide :

- des répétitives traditionnelles
- des structures répétitives proposées dans ce texte.

Le numéro d'octobre 1980 des "Communications of the ACM" fait référence aux travaux de J.D. WARNIER dans un article au titre prometteur : "LCS Logical Construction of Software". Quant au contenu, vous trouverez ci-après un résumé qui vous permettra d'en juger.

LOGICAL CONSTRUCTION OF SOFTWARE

DONALD R. CHAND AND SURYA B. YADAV
GEORGIA STATE UNIVERSITY

CONSTRUCTION LOGIQUE DE LOGICIELS.

INTRODUCTION.

Les phases de conception et d'implantation d'un logiciel sont souvent dans la pratique confondues, car : un programme est le produit fini du développement de logiciels et certaines méthodologies (cf YOURDAN, CONSTANTINE, MYERS) accentuent cette confusion.

Vis à vis du cycle de vie d'un logiciel, il est consistant, et nécessaire, de séparer la conception de la programmation.

Il ne faut pas appréhender un logiciel comme un système de programmes et sa documentation, mais comme un système d'ensemble de données et leurs traitements, puisque un programme consomme des entrées et livre des résultats. Cette nouvelle perspective précise le but de la conception logiciels : c'est l'identification et la définition des ensembles de données, de leurs traitements et de leurs connexités.

La "logical Construction of Software" (L.C.S.) doit faciliter les phases d'implantation, de maintenance et de validation.
"L.C.S." est une composition des méthodologies actuelles.

LA METHODOLOGIE "L.C.S."

Elle guide l'architecte par un ensemble de règles, de techniques, de procédures. On développe d'abord un modèle de systèmes d'information qui décrit le "quoi" (et ignore le "comment") en termes abstraits les données et leurs traitement. Ce modèle est construit récursivement à l'aide d'un modèle de systèmes ouverts (la frontière externe est appréhendée et les frontières internes déduites). L'addition d'un élément "contrôle" assure la validation du système à chaque étape récursive.

Puis on décrit les ensembles de données en groupes de fichiers à plat, afin d'obtenir une forme canonique aisément exploitable.
Les traitements de données sont décrits en termes de fonctions directement implantables, qui doivent être hiérarchisées à l'aide de la méthode Warnier.

Les connexités induites sont de trois types :

- entre les ensembles de données et les traitements de données;
- entre les ensembles de données eux-mêmes (matrice de préséance);
- entre les traitements eux-mêmes (diagramme de hiérarchie);

Ces trois types se déduisent du modèle de systèmes d'information.

L'intention est de mettre ainsi en valeur l'aptitude à être compréhensible et vérifiable.

On donne en figure 1 un exemple d'application.

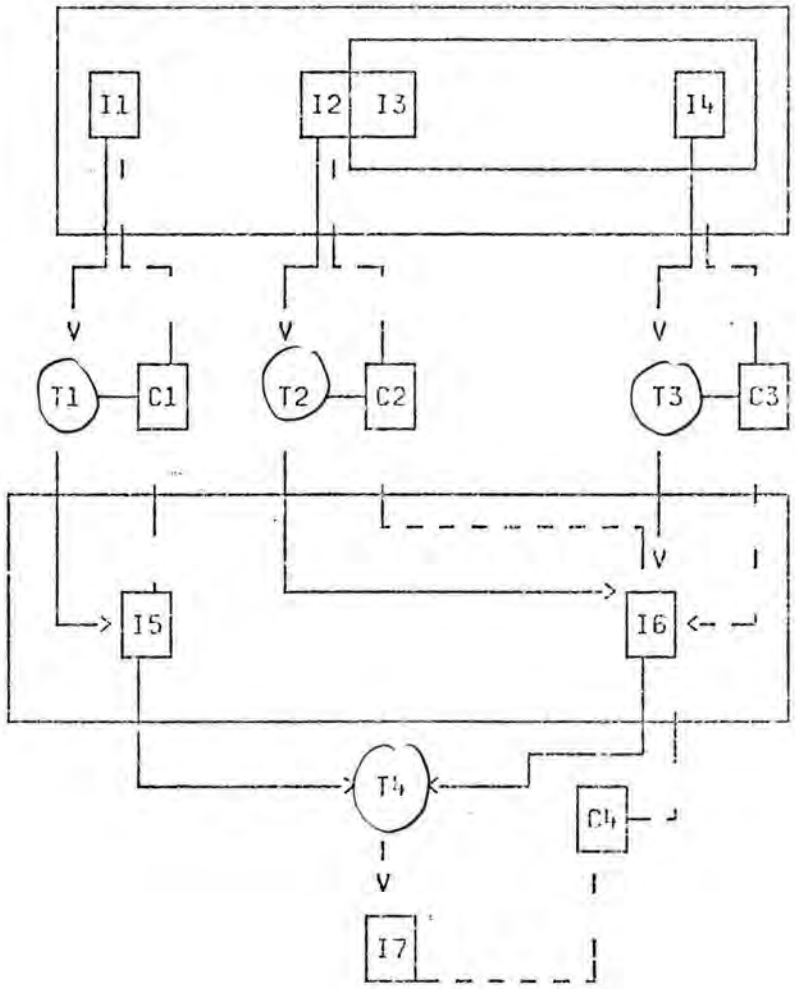
CONCLUSION.

"L.C.S." est une méthodologie d'aide à la conception de logiciels, à partir des besoins d'entrées/sorties, dans un formalisme explicitant les processus d'implantation et de maintenance.

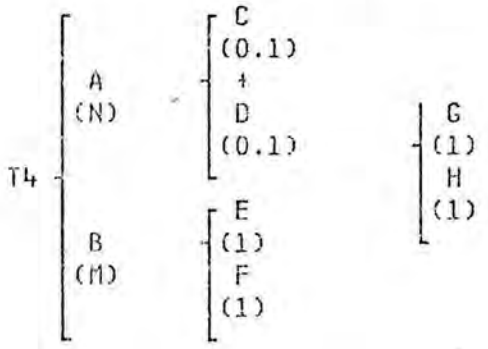
Cette capacité de description du produit permet d'exiger de ne réaliser l'implantation que lorsque la conception est complète et certifiée.

N.d.T. : il y a "L.C.S." et L.C.S. ...

a) Modèle de systèmes d'informations abstraites



b) Traitement de données T4 où les lettres A-H représentent les fonctions



c) Ensemble de données I6

I6F1(Key1,key1,item----)
I6F2(----------)

---

d) Connexité entre les ensembles de données

	I1	I2	I3	I4	I5	I6	I7
I5	I						-i
I6		I	I	I			-I

c) Connexité entre les traitements de données

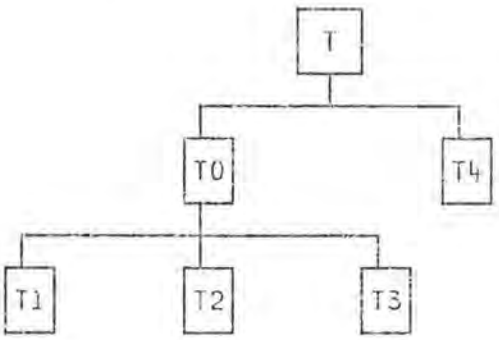


fig.1 Une description logique d'un logiciel en utilisant L.C.S

Commentaires :

a) est le diagramme du modèle de systèmes d'informations abstraites sous forme de réseau d'ensembles de données et de traitements de données

b) est la description logique en Warnier d'un traitement de données

c) est la description en fichier à plat d'un ensemble de données

d) est la matrice de préséance des ensembles de données

e) est l'arbre hiérarchique des traitements de données.

BONNES (OU MAUVAISES) FEUILLES .

Dans l'Informatique Nouvelle de décembre 1981 à la rubrique Informatique Plus est paru un article non signé intitulé "Sur la construction logique de Programmes".

L'auteur s'attaque à l'ouvrage de WARNIER "qui ne saurait servir de manuel d référence". Il en critique la présentation : "on a l'impression qu'il y manqu quelque chose", ce quelque chose "que le lecteur découvre enfin au chapitre I de la deuxième partie". L'auteur s'en prend aussi au titre de l'ouvrage "qu ne reflète guère le contenu" et au public à qui il s'adresse : "quant au choi du public il n'est pas indiqué".

Il attaque vigoureusement aussi la démarche L.C.P. qui, comme toutes le autres méthodes, "a pour objectif de neutraliser l'acidité contenue dans le problèmes", puis "Un outil des plus utiles dans certaines circonstances".

L'auteur nous dit que "L.C.P. n'est pas une méthode de conception des systèm au sens qu'en donne Constantine" en ajoutant "qu'il s'agit uniquement de con ception et non de construction des programmes".

Le processus de la démarche L.C.P. est ensuite tourné en dérision et surtou "WARNIER n'indique pas au lecteur dans quelles circonstances utiliser l méthode".

Une réponse aux attaques formulées sera rédigée par le bureau avec demand d'insertion.

LES INSTRUCTIONS PAR FAMILLES.

De la liste détaillée des Instructions par Familles (LIF) à la liste ordonné des Instructions par séquences (LIS).

INTRODUCTION.

Durant et aprés le cours LCP, la majorité des programmeurs abandonnent l'étud de la liste Détaillée des Instructions par Familles (LIF) pour quatre raison :

1. Aprés avoir conçu le programme en un ensemble de séquences logiques , i semble tout naturel ensuite d'étudier les instructions dans chaqu séquence logique donc de dresser la liste Ordonnée des Instructions pa Séquence (LIS).

2. Dresser la liste Ordonnée des Instructions par Séquence (LIS) a partir d la liste Détaillée des Instructions par Famille (LIF) est une tâche d recopie vraiment fastidieuse qui parait tout à fait inutile a programmeur.Cette liste par Famille lui semble ne servir à rien.

3. Même si les instructions par famille étaient écrites en COBOL, pa exemple, comme on ne peut pas les mettre directement à jour on ne voit pa l'intérêt de les étudier pour une seule et unique utilisation.

4. Le programmeur doit dresser trois listes d'instructions :

   - La liste Détaillée en Français par Famille
   - La liste Ordonnée en Français par Séquence
   - La liste Ordonnée en Cobol par Séquence

Ces trois listes imposant d'écrire la même instruction deux fois en franca et une fois en COBOL ne sont pas suffisamment motivantes pour être exécutée avec goût par les programmeurs qui n'étudient plus que la liste Ordonnée e COBOL des Instructions par Séquence (LIS).

COMMENT LES CONSERVER ?

La solution est simple. Il suffit de n'étudier que la liste Détaillée en COBOL des Instructions par Famille (LIF).

Il faut donc :

1. ECRIRE les instructions de chaque Famille en langage symbolique (BASIC, COBOL, ...) sur les imprimés prévus à cet effet en réservant une ou plusieurs feuilles par Famille.

2. RAJOUTER à chaque instruction :

   - Un numéro de Séquence logique (3 caractères)
   - Un numéro de Famille (1 c)
   - Un numéro d'instruction (2 c)

3. RANGER ces instructions par Famille en Bibliothèque.

4. RANGER tout ce qui est Description des Données du programme.

5. UTILISER une procédure dite LIFALIS qui triera les instructions par Séquence Famille Instruction, les éditera éventuellement et compilera le programme.

NUMERO DE FAMILLE	FAMILLES D'INSTRUCTIONS
1	LES COMMENTAIRES POUR SEQUENCES
2	LE NOM DES SEQUENCES
3	LES PREPARATIONS DE BRANCHEMENTS
4	LES CALCULS (ET LEURS PREPARATIONS)
5	LES SORTIES (ET LEURS PREPARATIONS)
6	LES ENTREES (ET LEURS PREPARATIONS)
7	LES BRANCHEMENTS
8	DIVERSES (READY TRACE - CALL MINUTEUR)

LE NUMERO D'INSTRUCTION :

A l'intérieur d'une séquence logique pour un calcul ou une sortie de résultats, il y a plusieurs instructions qui doivent impérativement se dérouler suivant un certain ordre. Il faut donc un numéro d'instruction dans la Famille et dans la Séquence.

L'EMPLACEMENT DES NUMEROS

- Le numéro de Séquence — Colonnes 1, 2, 3
- Le numéro de Famille — Colonnes 4

- Le numéro d'Instruction Colonnes 5,6

COMMENTAIRES

On peut aussi écrire des commentaires avec comme numéro 999999 *

COMMENT LES MODIFIER ?

LA liste des instructions par Famille étant en biblicthéque, il sera facile de la modifier pour corriger des erreurs de syntaxe et pour la mise au point du programme.

1. Les erreurs de syntaxe.

   Se corrigent facilement - L'emplacement d'une instruction dans la bibliothèque figure sur le listing après compilation.
   Si une erreur est apparue dans une instruction d'Entrée de Données en se reportant dans la liste par Familles à la Famille des Entrées on verra immédiatement l'instruction concernée ainsi que la même instruction si elle apparait dans d'autres séquences. On pourra ainsi corriger plusieurs instructions identiques.

2. La mise au point du programme.

   Si une remise à blanc d'une zone n'a pas été faite en sortie, en se reportant à la Famille des Sorties, on se rendra compte immédiatement :

   a) Si cette remise à blanc existe

   b) Si elle existe, a-t-elle été placée dans la bonne Séquence ?

   De même pour une remise à zéro d'une zone de calcul en se reportant à la Famille des Calculs, on pourra déceler l'erreur ou l'oubli sans avoir besoin de compulser tout le programme pour savoir si cette remise à zéro existe.

CONCLUSION.

Enseignants LCP vous êtes, j'en suis sûr, convaincus de l'intérêt de cette démarche nouvelle et originale qu'est l'étude des Instructions par Famille.

Elle permet d'avoir une vue globale du programme donc une meilleure lisibilité et prise de connaissance.

Il faut qu'elle trouve son prolongement par l'utilisation d'un logiciel simple à mettre en oeuvre qui incitera les programmeurs à ne pas l'abandonner.

Ce logiciel existe. Il est conversationnel. Il permet d'éditer ou non la liste des Instructions par Famille, de les trier et de compiler le programme. Ce logiciel est opérationnel sur IBM-34, IBM-4331. Il est à votre disposition si vous le désirez sur disquette.