MASTER EN SCIENCES INFORMATIQUES

Maintenance du système d'exploitation BS 2000
Outil de parcours de DUMPS

Marchal, Patrick

Award date:
1986

Awarding institution:
Universite de Namur

ANNEE ACADEMIQUE 1985-86.

MAINTENANCE DU SYSTEME D'EXPLOITATION BS 2000.

OUTIL DE PARCOURS DE DUMPS.

MEMOIRE PRESENTE PAR
MARCHAL PATRICK EN VUE
DE L'OBTENTION DU GRADE
DE LICENCIE ET MAITRE
EN INFORMATIQUE.

PROMOTEURS :

J. RAMAEKERS
P. DUMONT.

Nous tenons à remercier Mr. Ramaekers pour ses nombreux conseils méthodologiques et pour l'aide qu'il a bien voulu nous apporter tout au long de cette année.

Nous tenons également à remercier tout spécialement Mrs. P. Dumont et B. Bolle pour leur aide efficace ainsi que pour leurs idées et leurs conseils prodigués tout au long de l'élaboration de ce travail qui n'existerait pas sans eux.

Enfin, nous tenons à remercier tout le personnel de Siemens Software de Rhisnes, ainsi que les membres du team " Device Management " de Siemens Munich et plus particulièrement Mr. Kulzer, pour leur accueil et leurs conseils lors de l'implémentation de ce travail et lors de mon stage.

TABLE DES MATIERES.

INTRODUCTION

La maintenance est le fait d'entretenir un matériel ou de recompléter des unités de combat, telle est la définition de ce mot donnée par le " Petit Robert ". Mais en informatique, définir ce terme avec précision, n'est pas chose aisée. On s'accorde généralement à dire que la maintenance d'un logiciel est l'ensemble des changements apportés à des programmes opérationnels dans le but de les garder opérationnels.

Cependant, ce terme reste assez flou par la diversité des actions qu'il regroupe. C'est pourquoi certains auteurs distinguent plusieurs types de maintenance tels que la maintenance corrective (correction d'erreurs découvertes après que le programme soit devenu opérationnel) et la maintenance adaptative (adaptation des programmes à des changements externes; compilateur, hardware, ...) (Grem84). Parikh mentionne également la maintenance perfective (amélioration de fonctions) (Pari86).

D'autre part, les logiciels sont souvent livrés tardivement et, habituellement, ne sont pas fiables (Zelk78). Il n'est donc pas rare que des programmes soient utilisés alors qu'ils ne fonctionnent pas correctement (Pari86).

Comment, dans ce cas, peut-on discerner la maintenance du développement ? Ne s'agit-il pas de la mise au point et de la continuation du développement ? Lorsqu'on apporte des modifications à un logiciel, peut-on encore parler de maintenance ? Ne s'agit-il pas d'un développement continu ?

On le voit, le problème de la définition de la maintenance n'est pas simple, mais le but de ce travail n'en est pas la résolution. Nous laisserons donc au lecteur désireux d'approfondir le sujet, le soin de consulter les auteurs déjà cités et pour la suite de ce travail, nous garderons au terme " maintenance " son caractère général.

La maintenance est une étape importante du cycle de vie d'un logiciel, car, même après sa mise au point et sa distribution, un programme recèle encore des erreurs plus ou moins nombreuses. En fait, la maintenance d'un programme représente 67% de son coût total (Vlan84), et la correction des erreurs y occupe une part non négligeable. Cette étape constitue une lourde charge pour les entreprises; on estime que la plupart d'entre elles y consacrent 50% de leur budget informatique et que plus de 30 billions de dollars sont dépensés chaque année à cette fin (Pari86). Il est donc compréhensible que l'on cherche à réduire ces coûts.

En faisant abstraction des efforts à fournir, liés au développement des logiciels (plus de fiabilité, meilleurs tests,...), le problème consiste à découvrir rapidement les erreurs, à les corriger et à faire circuler ces corrections afin que, si ces erreurs se représentent à nouveau, on ne perde plus de temps à les solutionner. Si, il y a une dizaine d'années, ce problème n'était pas trop préoccupant , il devient aujourd'hui crucial étant donné la taille sans cesse croissante des logiciels et le nombre toujours plus important des développeurs concernés par ceux-ci (à titre d'exemple, citons 5000 hommes-année pour un système d'exploitation (Zelk78)). Il semble donc impératif, pour essayer de diminuer les coûts de la maintenance, de développer un ensemble d'outils qui permettent d'améliorer ce travail en assurant une correction plus rapide et une meilleure circulation de celle-ci.

Dans ce travail nous nous intéresserons plus spécialement à la maintenance de programmes systèmes et en particulier du système BS 2000 de Siemens.

1. DEFINITION DU PROBLEME

1.1 La situation actuelle.

Essayons de présenter la démarche suivie par les personnes chargées de la correction afin de mieux discerner les étapes où l'on pourrait intervenir.

Lorsque le système ne peut plus continuer, lorsque " il s'est planté " comme nous le disons familièrement, c'est une erreur qui se manifeste (erreur d'adressage dans le code du système, tentative d'exécution d'un ordre qui n'est pas un code opératoire, ...). On se trouve donc confronté à un crash du système ou d'une partie de celui-ci. Il est nécessaire alors de créer un " dump du système ", c'est-à-dire établir un listing qui sera le reflet, la copie lisible du contenu de la mémoire ou d'une partie de la mémoire du système. Généralement cette copie sera présentée sous la forme hexadécimale.

Dans la pratique, les dumps sont d'abord créés sur un support intermédiaire, bande magnétique ou disque, puis édités et imprimés. Il faut alors parcourir ce listing pour vérifier le contenu des données qui sont organisées en tables. Mais on ne connaît malheureusement pas leur adresse - excepté pour la première - puisqu'elles sont créées dynamiquement au fur et à mesure des besoins.

La méthode de parcours du listing est la suivante: chercher dans la première table, un pointeur vers une table qui possède un pointeur vers une autre table, qui possède à son tour un pointeur vers une table qui ... jusqu'à trouver un pointeur vers la table qu'il fallait atteindre. C'est donc en se déplaçant de table en table à l'aide de pointeurs que l'on trouvera la table cherchée.

Lorsque cette table est localisée, il faut vérifier si certaines valeurs ont été ou n'ont pas été modifiées et la modification de ces valeurs permettra de déterminer si oui ou non le programme " est passé " par ce point. Il faut alors recommencer la recherche d'autres tables pour effectuer les mêmes observations. Ainsi, petit à petit, se repéreront les parties du programme exécutées et celles qui ne l'ont pas été. C'est en analysant ces traces de passage que nous pourrons essayer d'établir la raison de cette erreur.

Il restera à la corriger et à diffuser la correction parmi toutes les versions existantes du programme. En effet, si seule la correction d'une version était réalisée pour chaque erreur découverte, il faudrait faire face à une croissance explosive du nombre de versions différentes d'un même programme.

Par exemple, supposons qu'un programme A soit installé sur trois sites I,II,III et qu'une erreur survienne sur le site I. Si le système du premier site est le seul à être corrigé, celui-ci possédera un système A' tandis que les sites II et III garderont un système A inchangé.

Supposons encore qu'une erreur se produise sur les sites I et II, comment déterminer si cette erreur est la même sur les deux sites puisque les systèmes sont différents? La correction de celle-ci nécessitera donc la modification des deux systèmes. Les trois sites posséderont trois programmes différents: A'' sur I, A''' sur II et A sur III (Zelk78).

Ce problème est évité par l'emploi de bases de données contenant les modifications apportées aux différents programmes permettant ainsi la diffusion des corrections de manière plus aisée.

1.2 Critiques de la situation traditionnelle et mesures possibles.

L'archaïsme de la méthode se révèle immédiatement et on imagine aisément que ce travail consistant à tourner et retourner des pages de listings, à comparer des tables, est long, lent, et fastidieux. De plus, la hauteur d'une pile de listings pour un dump peut actuellement avoisiner le mètre et cette quantité ne cesse d'augmenter.

En outre, pour parcourir ce listing, la connaissance de la structure de toutes les parties du système est nécessaire. Or, la tendance actuelle est d'encapsuler les diverses parties de ce système pour éviter que les développeurs d'un sous-ensemble ne doivent tenir compte des structures des autres parties. Nous pouvons donc supposer qu'il sera bien vite impossible de continuer de cette manière.

Mesures possibles.

La première chose qui vient à l'esprit est de se demander s'il ne serait pas possible de faciliter ce déplacement dans un dump et cette comparaison de tables.

Plusieurs phases peuvent être distinguées pour aboutir à la réalisation d'un programme qui atteint ces objectifs:

- Afin de pouvoir se déplacer dans un dump, notre logiciel devra connaître ce que le dump représente, ce qu'il contient. Pour cela, il faudra définir un langage de description de la structure du système et ensuite décrire le système au moyen

de ce langage.

- L'étape suivante sera la réalisation de l'outil de parcours proprement dit. Cet outil devra permettre de se déplacer dans une masse de données en utilisant la description du système mentionnée ci-dessus.
- Il serait agréable pour l'utilisateur de pouvoir analyser le dump à partir du terminal, c'est pourquoi un outil qui permet, par exemple, de connaître le contenu d'une table ou de savoir si un élément de celle-ci a été ou n'a pas été modifié, constituera une autre étape.
- Pour permettre à l'utilisateur, soit d'interrompre un travail d'analyse et de le reprendre plus tard, soit de conserver les symptômes de l'erreur (éléments de tables modifiés ou non, ...), il sera nécessaire d'écrire un nouvel outil.
- Nous disposerons alors d'un certain nombre de symptômes pour chaque erreur. Nous pourrons écrire un outil qui, en présence d'une erreur et à l'aide des symptômes des erreurs précédentes et de l'outil de parcours, pourra détecter si celle-ci n'est pas une erreur déjà connue.
- Pour gérer tous ces symptômes propres à chaque erreur, un autre outil sera également nécessaire.

Cet ensemble d'outils devrait donc permettre à l'utilisateur de vérifier rapidement si l'erreur est nouvelle ou non. Dans le cas d'une erreur non encore rencontrée, l'outil de parcours permettrait de se déplacer plus facilement dans le dump et donc accélérerait sa découverte ainsi que celle de ses symptômes.

Par sa taille et sa complexité, cet ensemble d'outils dépasse largement le cadre d'un mémoire. Nous nous attacherons à la réalisation de l'outil de parcours du dump. La définition d'un langage de description de la structure du système ainsi qu'une partie de la description elle-même de ce système ont déjà été réalisées par B. Bolle (Boll84).

1.3 Les hypothèses.

Il est important de rappeler que nous nous intéressons à des logiciels de base bien que cet ensemble d'outils puisse éventuellement être adapté et utilisé pour d'autres logiciels de taille importante.

1.4 La démarche.

Après cette présentation du problème, nous essayerons de passer rapidement en revue quelques outils qui existent déjà, nous tenterons de détecter les raisons pour lesquelles ils ne répondent pas à notre attente. Ensuite, après un rappel de la représentation de la structure de données, définie par B. Bolle, (Boll84) nous décrirons le programme de parcours proprement dit. Dans un quatrième temps, nous essayerons d'établir les limites de ce programme ainsi que les extensions et les améliorations qui pourraient y être apportées.

2. QUELQUES OUTILS EXISTANTS

2.1 Introduction.

Le but de cette partie est de donner un aperçu des quelques outils d'aide à la maintenance de systèmes de base existants chez Siemens. Nous essayerons de déterminer quelles sont leurs fonctions ainsi que leurs défauts.

Dans une seconde partie, nous décrirons les particularités du système de trace que nous avons eu l'occasion d'améliorer lors de notre stage chez Siemens à Munich.

2.2 Quelques outils d'accès aux dumps de systèmes de base.

Parmi les outils qui permettent d'accéder aux dumps citons A.I.D. (Algorithm for Interactive Debugging) qui est un outil de test interactif et indépendant pour le système BS2000 (Kais83, Hirs86), DAMP (Dump Analysis and Maintenance Program) (Damp), Helga (Deco82) et Marion (Dumo85).

Les principales fonctions de ces logiciels sont de permettre à l'utilisateur l'accès aux adresses de la mémoire virtuelle, aux tables systèmes, aux fichiers dumps, à la mémoire du système, aux adresses réelles. Ils permettent également l'affichage de toute partie de la mémoire accessible, la modification de la zone mémoire du système et la recherche d'une suite de caractères dans la mémoire virtuelle ou dans un fichier dump. Certains, comme Helga, donnent à l'utilisateur la possibilité de lister les tâches appartenant à une queue (file d'attente d'entrées-sorties, ...) ou une catégorie (système, dialogue, batch, ...).

Ce qui différencie ces outils, c'est la méthode de chargement dans le système et la manière par laquelle ils accèdent aux dumps.

Que leur manque-t-il ?

Dans l'ensemble, ces outils ne sont pas satisfaisants pour la maintenance d'un logiciel car ils ne connaissent qu'une partie du système. Ils ne permettent l'interprétation que de certaines tables et pour le reste du système, ils ne peuvent présenter qu'une masse brute de données. C'est alors à l'utilisateur, de connaître la structure et d'interpréter ces données.

De plus, la taille de ces outils est déjà importante et leur extension à l'ensemble du système aboutirait rapidement à un volume comparable à celui du système lui-même.

Ces applications sont programmées, ce qui implique que toute modification du système entraîne la modification des outils et l'obligation de les tester à nouveau. Ils ne sont donc généralement opérationnels que bien longtemps après les modifications du système et ne peuvent servir à la mise au point de celles-ci.

C'est à ces manquements qu'essaie de répondre l'ensemble d'outils développés dans ce mémoire.

2.3 " Trace dump ".

L'outil " trace dump " est un outil propre au team " Device Management " dans lequel j'ai travaillé lors de mon stage à Munich. C'est pourquoi je le présenterai plus particulièrement ci-dessous. Il permet d'enregistrer des traces provenants de différents modules, traces qui, en cas d'erreur, donnent plus de facilité pour déterminer l'origine de celle-ci.

2.3.1. Le principe.

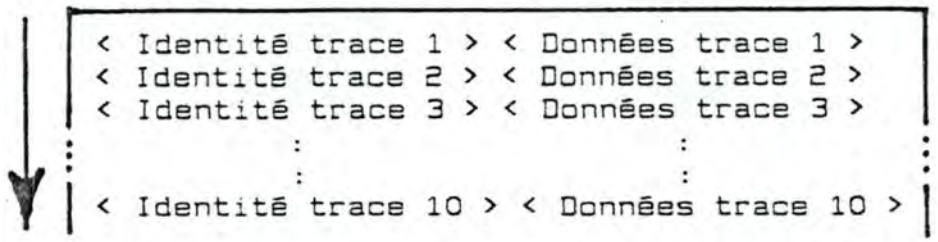
Cet outil est présent en permanence dans le système et il peut être actionné dynamiquement, c'est-à-dire qu'il n'est pas nécessaire de stopper le système pour le mettre "on" ou "off", ou pour choisir de quel(s) module(s) seront enregistrées les traces.

Chaque module, lors de l'entrée dans celui-ci et également lors de la sortie, envoie des informations au module trace dump qui, suivant ce qui lui a été demandé, décide de traiter ou non les données reçues. Si un traitement des informations de ce module lui a été demandé, il va inscrire la trace dans une zone mémoire qui a été préalablement réservée lors de l'initialisation du système. Les écritures se font les unes à la suite des autres et de manière circulaire, c'est-à-dire que lorsqu'on arrive en bas de la zone, les écritures recommencent au début (fig 2.1). De plus, chaque trace possède un indicateur qui mentionne s'il s'agit d'une trace normale (normal trace) ou d'une trace d'erreur (error trace) envoyée lorsque se produit une erreur (réservation d'un " device " non existant, libération d'un " device " non réservé, ...).S'il s'agit d'une " error trace ", celle-ci sera également inscrite dans une table des erreurs qui a aussi été réservée à l'initialisation. Cette table permet d'enregistrer environ une dizaine d'erreurs car on estime qu'au delà de ce nombre, celles-ci ne sont plus significatives et que de toute façon un crash système a de grandes chances de se produire avant d'atteindre ce nombre.

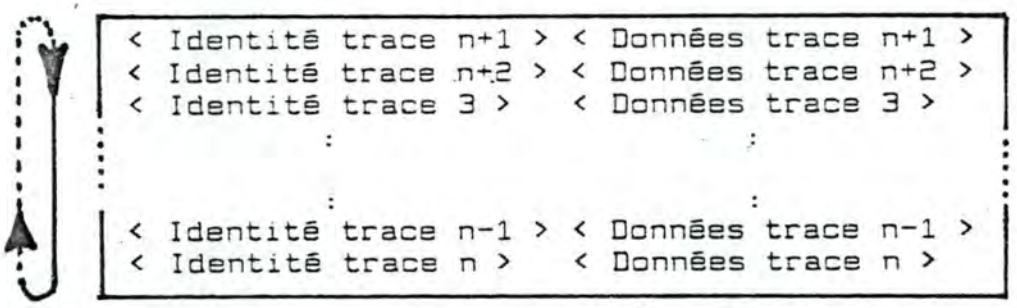
Ainsi, lors de l'analyse d'un crash, on dispose d'une table avec l'historique de tous les évènements survenus dans les dernières minutes ainsi que d'une table avec le récapitulatif des erreurs (fig 2.2). Cela permet de resituer ces erreurs dans leur contexte et de suivre pas à pas ce qui s'est passé afin d'en déterminer les raisons. On peut en quelque sorte, comparer cet outil aux boîtes noires qui équipent les avions actuellement.

L'expérience a montré, qu'à l'aide de ce système, il était parfois possible en corrigeant une erreur, de détecter un autre défaut du module et de permettre sa correction avant même qu'il ne provoque une erreur.

Cet outil offre donc l'avantage de rendre la correction des erreurs plus rapide et plus aisée, mais aussi il permet d'accélérer la mise au point par la détection de problèmes avant qu'une erreur ne se produise.

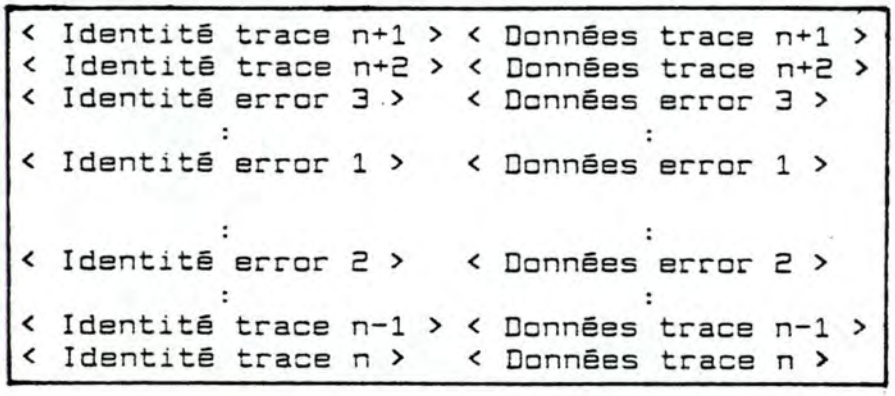


Les traces sont inscrites les unes à la suite des autres.

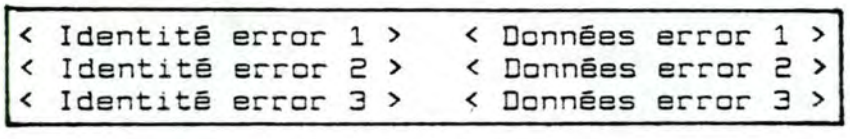


Lorsque le bas de la zone est atteint, la trace suivante est écrite en haut de celle-ci.

Fig 2.1



La table avec l'historique des événements.



La table avec le récapitulatif des erreurs.

Fig 2.2

2.3.2. Contribution personnelle.

Lors de mon stage, j'ai eu l'occasion de travailler sur cet outil et d'y apporter quelques modifications dont les principales sont décrites ci-après.

A. Création de " pages historiques ".

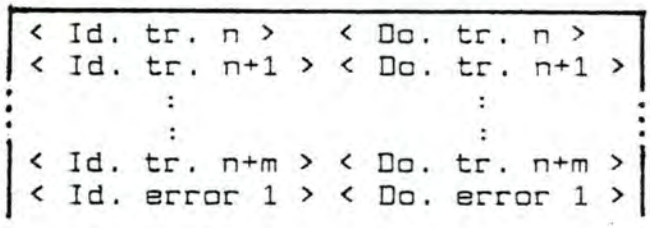
Les traces normales, comme nous l'avons dit ci-dessus, sont inscrites dans une page mémoire, réservée lors de l'initialisation du système. Cette méthode a deux inconvénients :

- Ainsi que nous l'avons vu plus haut, le module de trace peut être activé ou désactivé dynamiquement, ce qui veut dire que la page mémoire peut rester vide parce que le système n'est pas activé. Elle est donc réservée inutilement.

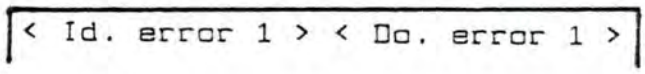
- Le système est très coûteux en terme de place mémoire car la zone réservée est très grande (±60k)

Nous avons donc cherché à améliorer ces deux points: d'une part, réduire la taille de la zone et, d'autre part, effectuer la réservation à la demande. Mais il fallait garder un espace suffisant pour conserver un nombre significatif de traces afin de pouvoir resituer les erreurs dans leur contexte.

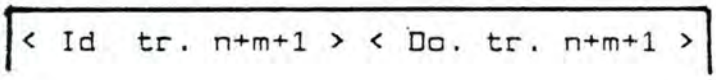
Nous avons donc opté pour la solution suivante : La taille de la zone a été ramenée à 254 entrées de 32 bytes. Lors de l'activation du trace dump, la réservation d'une zone, dans laquelle vont s'inscrire les traces, est demandée au système. L'écriture de ces traces se fait de la même manière que celle décrite plus haut: en fin de zone, l'écriture recommence à partir du début. Tant que le trace dump reçoit des traces normales, celles-ci sont inscrites dans cette zone. Lorsqu'une trace d'erreur se manifeste, elle s'inscrit dans la zone ainsi que dans la table des erreurs, tout comme dans l'ancienne méthode. Mais une fois ce travail exécuté, il faut vérifier s'il s'agit d'une nouvelle erreur, c'est-à-dire d'une erreur qui n'avait pas encore été enregistrée dans la table des erreurs car, dans ce cas, le module trace dump demande une nouvelle page au système. Celle-ci devient la page courante dans laquelle s'écriront les traces futures, tandis que l'ancienne page courante est conservée; cette page contient en fait l'historique de la première erreur (fig 2.3).



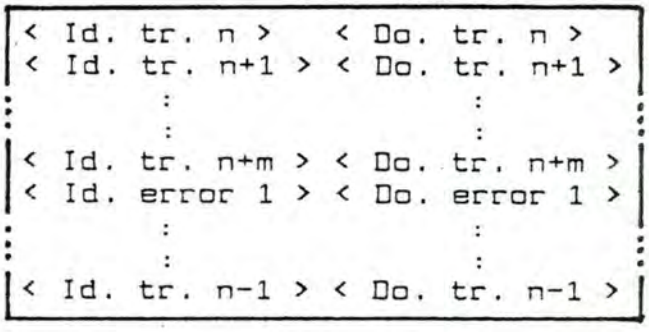
Lorsqu'une nouvelle error trace se produit, elle est inscrite dans la page courante ...



... ainsi que dans la table des erreurs ...



... et on change de page,



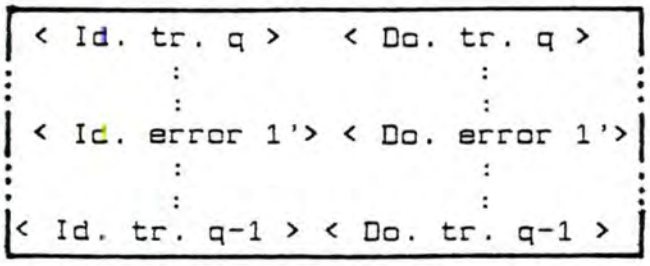
tandis que l'ancienne page courante devient une page historique.

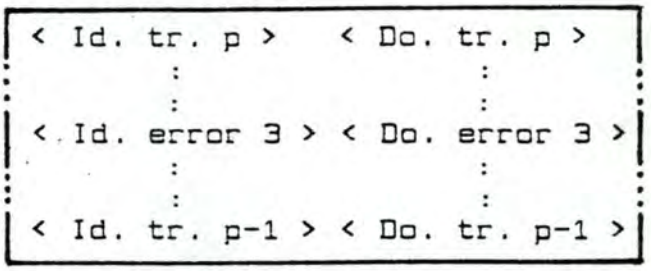
Fig 2.3

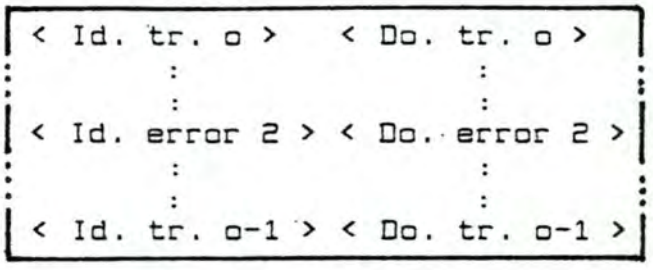
Par contre, s'il s'agit d'une erreur déjà présente dans la table des erreurs, aucune nouvelle page n'est demandée et les traces continuent de s'inscrire dans la page courante (fig 2.4).

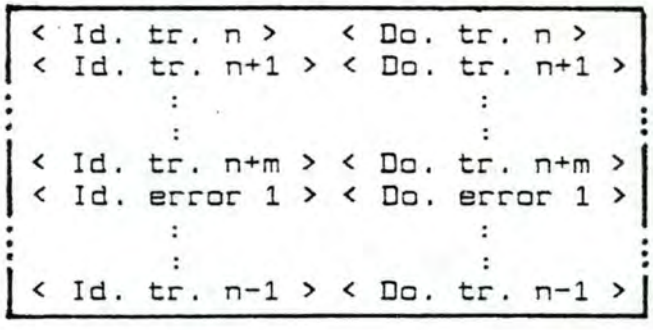
De cette manière, il est possible de sauver huit pages " historique d'erreur ", après quoi il n'y aura plus de changement de page. Ce nombre peut-être modifié dynamiquement mais comme nous l'avons signalé plus haut, il est généralement admis que les erreurs suivantes ne sont plus significatives, c'est pourquoi une valeur par défaut de huit a semblé suffisante.

Cette technique permet d'éviter le gaspillage en ne réservant que la place nécessaire au fur et à mesure de son utilisation.









Lorsqu'une error trace déjà enregistrée se produit, elle est inscrite dans la page ainsi que dans la table des erreurs. Mais les traces continuent à s'inscrire dans la même page courante.

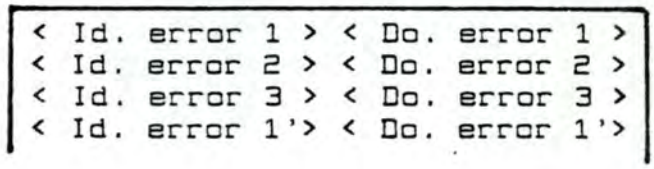
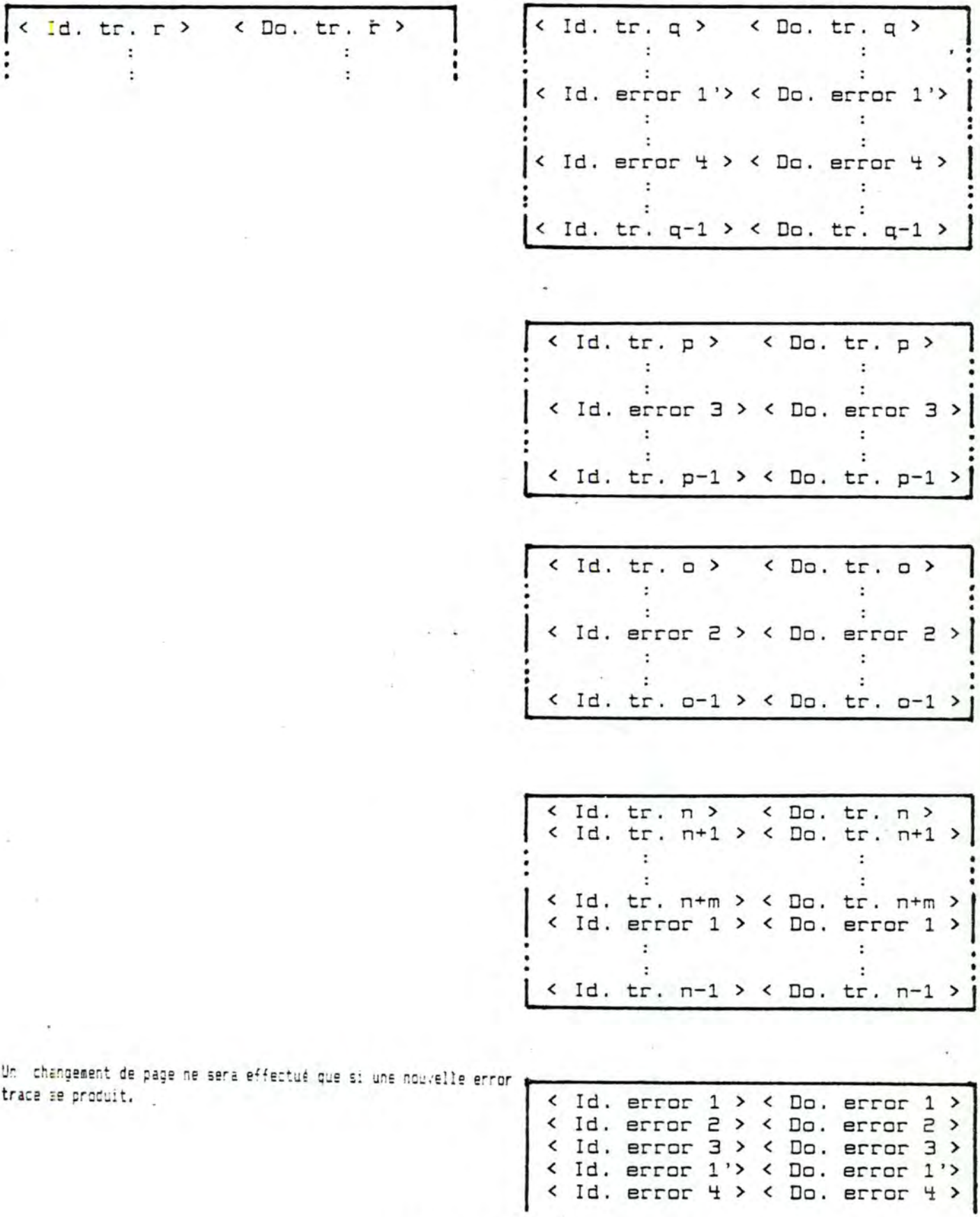


Fig 2.4



Un changement de page ne sera effectué que si une nouvelle error trace se produit.

Fig 2.4 bis

3. Présentation des tables historiques.

Le changement de méthode nécessite une adaptation de l'affichage de ces résultats, nous avons donc adapté le module de présentation des tables obtenues de la manière suivante :

- La ligne indiquant l'adresse de début, de fin et l'adresse courante pour la table des erreurs ainsi que pour la page courante a été conservée . Une ligne mentionnant le nombre de pages historiques existantes a cependant été ajoutée (fig 2.5.).

```
$$$$$$$$$$ T R A C E D U M P $$$$$$$$$$

TRACE POINTER      ANFANG     ENDE       AKTUELL
NDM-ERROR-TRACE    00079C8C   0007AC8C   00079E0C
NDM-ABLAUF-TRACE   00324020   00325FE0   00324260
NDM # HISTORICAL PAGE(S) :  0005
```

Fig 2.5

- Il nous a semblé intéressant, pour la facilité du lecteur, de lui indiquer dans quelle page historique se trouve l'erreur qu'il souhaite examiner. C'est pourquoi lors de l'écriture d'une nouvelle erreur dans la table des erreurs, un commentaire signalant le numéro de page historique est également inscrit (fig 2.6).

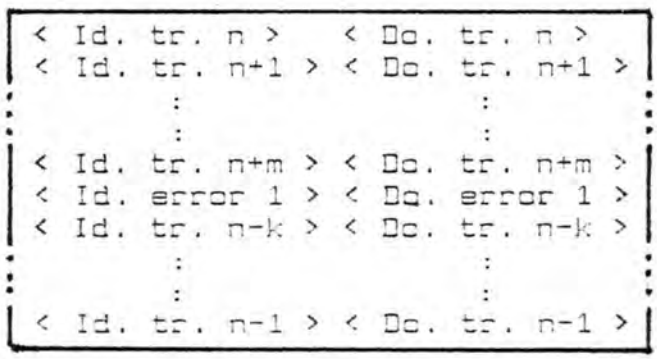
```
N D M - E R R O R - T R A C E

RMPRINVL.............XX........
RMPRINVL ON HISTORICAL PAGE : 01
RMPRINVL.............VB........
RMPRINVL ON HISTORICAL PAGE : 02
RMPRINVL.............XX........
SMSG0030....PID: 00 ECX: 0000000
0 ERA: 00000000 ..................
SMSG0030 ON HISTORICAL PAGE : 03
RMPRINVL.............VB........
RMPRINVL.............CC........
RMPRINVL ON HISTORICAL PAGE : 04
RMPRINVL.............AA........
RMPRINVL ON HISTORICAL PAGE : 05
.....................................
```

Fig. 2.6

- L'édition de la page courante n'a pas été modifiée puisque, à part la longueur, le principe reste inchangé.

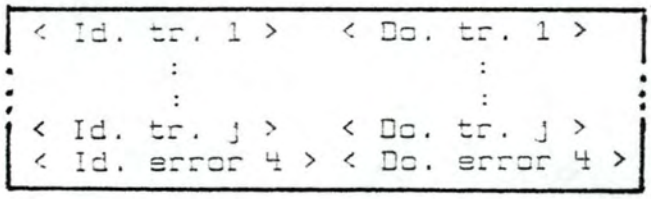
- La présentation des pages historiques par contre, a dû être créée puisque ces pages n'existaient pas auparavant. Afin d'aider le lecteur au maximum, nous avons décidé d'imprimer ces tables dans l'ordre anti-chronologique, c'est-à-dire en partant de la position courante au moment du changement de page - correspondant à l'écriture de la trace d'erreur - et en remontant jusqu'au début de la page . L'impression recommence au bas de la page et se poursuit jusqu'à la position de départ (fig 2.7). Il est évident que si la page n'a pas été complètement remplie, c'est-à-dire si la portion entre la position courante et la fin de cette page est vide, cette partie n'est pas imprimée.



N D M H I S T O R I C A L T R A C E P A G E # 0001

< Id. error 1 > < Do. error 1 >
< Id. tr. n+m > < Do. tr. n+m >
< Id. tr. n+1 > < Do. tr. n+1 >
< Id. tr. n > < Do. tr. n >
< Id. tr. n-1 > < Do. tr. n-1 >
< Id. tr. n-k > < Do. tr. n-k >

Fig 2.7



N D M H I S T O R I C A L T R A C E P A G E # 0004

< Id. error 4 > < Do. error 4 >
< Id. tr. j > < Do. tr. j >
< Id. tr. 1 > < Do. tr. 1 >

Fig 2.7 bis

3. LE PROGRAMME DE PARCOURS

3.1. Introduction.

Notre contribution à l'ensemble d'outils décrits précédemment est la réalisation d'un logiciel capable de rechercher une table dans un dump à l'aide de la description de la structure de données de celui-ci. L'idée générale, est de se servir de cette représentation comme d'une carte géographique du dump afin de trouver un chemin qui conduit à la table désirée.

L'avantage d'un tel outil, par rapport aux autres logiciels d'accès aux dumps, est son indépendance vis-à-vis du dump sur lequel il travaille. En effet, dans le cas des autres outils (A.I.D., DAMP, ...), chaque fois que la structure de données change, il faut reprogrammer tout l'outil. Avec ce nouveau logiciel, il suffit de modifier la description de la structure de données (ce qui peut être fait en recompilant avec les nouvelles " dsects " des tables) pour qu'il redevienne opérationnel. Avantage donc, non seulement pour la facilité de mise à jour, mais aussi pour la réduction de la période d'indisponibilité.

Après un rappel de la description de la structure de données, nous établirons les hypothèses du travail avant d'aborder l'outil de parcours lui-même.

3.2 Description de la structure de données.

Dans cette partie nous allons rappeler la représentation de la structure de données telle qu'elle a été définie par B. Bolle (Boll84). Toutefois, le langage de description ne nous concerne pas ici; nous nous limiterons donc, après la présentation de la structure de données, au résultat de la déclaration de cette structure puisque c'est à l'aide de ce résultat que nous travaillerons.

3.2.1. Les données.

A. Les tables.

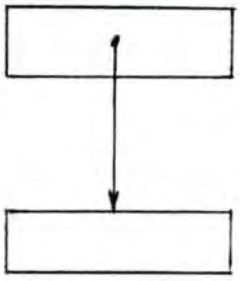
Les données qui définissent l'état du système (l'état des tâches et des ressources) sont contenues dans des tables. A chaque tâche existant dans le système, correspond une table contenant des informations sur des points tels que l'état ou le type de la tâche par exemple,. Chaque fichier est également décrit par une table. Ces différentes tables sont reliées entre elles par des liens formant ainsi un réseau.

Certaines tables sont uniques (telle la XVT qui contient les informations nécessaires pour le contrôle de toutes les tâches existantes). D'autres par contre, existent en plusieurs exemplaires (telles les TCB qui décrivent les tâches) et sont alors reliées les unes aux autres par une ou plusieurs structures de données que nous appellerons liens.

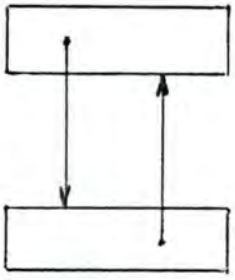
B. Les liens entre les tables

Il faut faire une distinction entre les liens qui relient des tables multiples entre elles et les autres types de liens.

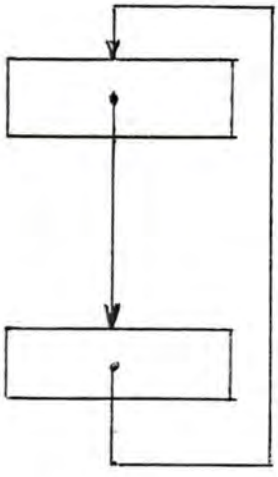
Les tables multiples sont reliées les unes aux autres. Si une table à laquelle on accède, peut aussi donner accès à la table d'où l'on vient, elle portera le nom de queue double. Si au contraire, elle ne peut pas accéder à la table de départ, elle portera le nom de queue simple. La dernière table de la file peut également être reliée à la première, si l'accès peut se faire dans les deux sens nous parlerons d'anneau double, ou d'anneau simple dans le cas contraire.



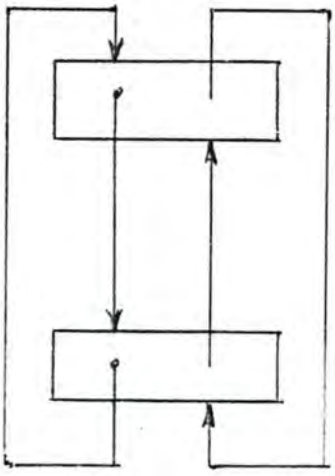
Queue simple.



Queue double.



Anneau simple.



Anneau double.

Pour les tables de types différents, il existe trois types de liens: le pointeur, le pointeur multiple et le vecteur.

- Le pointeur est le lien le plus courant entre deux tables, il est aussi le plus simple: la première table contient l'adresse de la deuxième.

Ce pointeur peut être de types différents: soit une adresse simple avec une longueur de 3 ou 4 bytes, soit un déplacement sur deux bytes ou encore la longueur d'une zone. Dans ces deux derniers cas, il faut ajouter le déplacement ou la longueur à l'adresse du début de la table courante pour trouver la suivante.

- Le pointeur multiple est un lien spécial. Il peut donner accès à des tables ayant un type différent suivant la situation dans laquelle elles sont employées. Par exemple, le buffer d'un fichier peut-être considéré comme une table mais il peut contenir différents types de structure de données suivant le type du fichier (data-block ISAM ou block SAM). Donc, la table sera également d'un type différent suivant le fichier.

- Le dernier type de lien entre les tables est le vecteur. Une table peut être reliée à plusieurs autres tables, elle contiendra alors un vecteur dont les composants sont des pointeurs vers toutes ces différentes tables.

Remarque : Certains pointeurs ont un inverse. C'est-à-dire un pointeur qui va de la table destination à la table origine.

C. Pages partielles

Nous avons défini les tables et les liens entre ces tables. Cependant le système peut allouer des pages partielles à une tâche. Les données relatives à la gestion de ces pages ne correspondent pas aux types définis précédemmment. Il nous faut donc considérer un nouveau type de données que nous appellerons " Partial Page ". Celui-ci contient trois sous-types correspondant chacun à une classe de mémoire du système (classe 3, classe 4, classe 5) .

3.2.2. Le graphe formel.

Il apparaît que toutes ces tables et tous ces liens manipulés par le système, forment un graphe. Nous allons dès lors établir un graphe formel décrivant ce graphe physique et définir des symboles de représentation de ces différents éléments.

A. Les sommets du graphe formel

Les tables, les queues simples, les queues doubles, les anneaux simples, les anneaux doubles et les " partial page " constitueront les sommets du graphe formel. Ils seront représentés par un rectangle dans lequel sera inscrit le nom de la structure correspondante.

Afin de pouvoir distinguer les différents types de noeuds, le nom des queues simples sera précédé du symbole %, le symbole %% précédera le nom des queues doubles, tandis que celui des anneaux simples sera précédé du symbole # (## pour les anneaux doubles). Pour les " partial page " on utilisera PP3, PP4, PP5 selon le sous-type considéré.

Remarque : Par la suite nous désignerons tous ces éléments par le terme de noeud.

B. Les arcs du graphe formel.

- Référence simple.

Lorsqu'un noeud contient un pointeur vers un autre noeud, nous représenterons ce pointeur par le symbole >

- Référence vecteur.

Un vecteur de pointeurs d'un noeud vers un autre noeud sera représenté par le symbole >>

- Référence multiple.

Si un noeud contient un pointeur multiple vers d'autres noeuds, la référence multiple sera représentée par un arc ayant plusieurs extrémités.

- Relais de composition.

Une liste (queue ou anneau) est toujours composée d'un certain type de noeud. Le symbole >< représentera ce type de lien.

- Racine du graphe formel.

Le graphe formel a toujours une racine qui est un point d'entrée dans le graphe. En général, la table XVT est cette racine.

- Elément NIL.

Lorsqu'un pointeur n'a pas d'extrémité, on considérera qu'il pointe vers un noeud fictif appelé NIL.

3.2.3. Exemple de graphe formel.

Représentons le graphe suivant :

- La racine du graphe est la XVT. Cette table possède un pointeur vers la TLT et un vecteur dont les composants sont des pointeurs vers des queues doubles appelées QTCB.

- La table TLT possède une référence vecteur vers des tables appelées TCB.

- Les queues doubles QTCB sont composées de TCB.

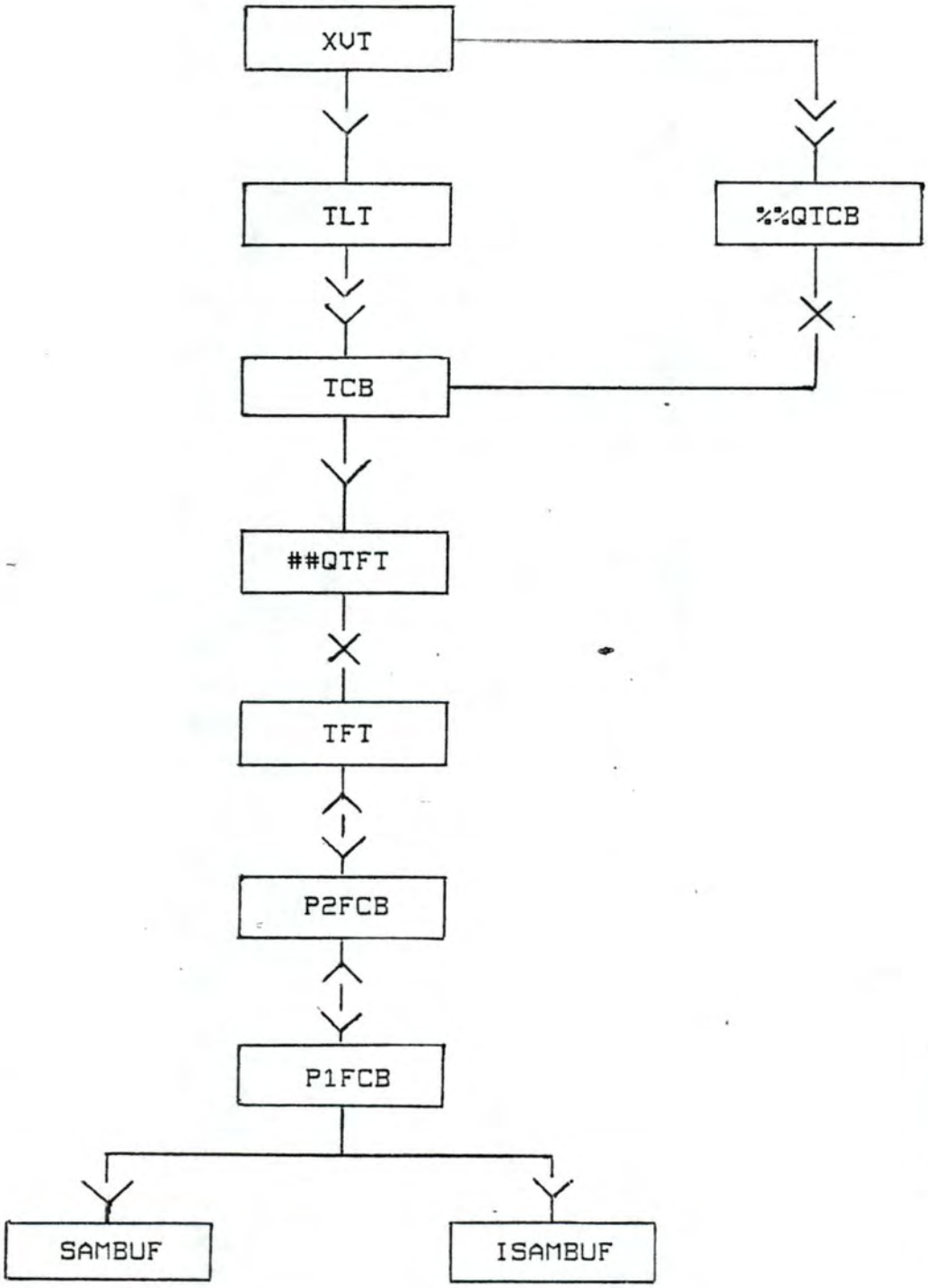
- La TCB possède une référence vecteur vers des queues doubles dont le nom est QTFT.

- Chaque QTFT est composée de tables de nom TFT.

- La TFT est liée par un pointeur (doté d'un inverse) à une table nommée P2FCB.

- Cette table possède un pointeur (avec un inverse) vers une table P1FCB.

- La P1FCB contient un pointeur multiple vers les tables SAMBUF et ISAMBUF.


XVT
TLT
%%QICB
TCB
##QTFT
TFT
P2FCB
P1FCB
SAMBUF
ISAMBUF

3.2.4. Représentation physique du graphe.

Après avoir défini ce graphe formel, il faut le représenter physiquement pour qu'il puisse être utilisé par l'outil de parcours.

Globalement, on peut dire que chaque noeud et chaque arc du graphe sont représentés par un descripteur et qu'un dictionnaire répertorie tous les noeuds existants.

- Chaque entrée du dictionnaire reprend le nom d'un noeud ainsi que son adresse.

- Le descripteur d'un noeud contient l'adresse de l'entrée correspondante dans le dictionnaire et les adresses des différentes listes de références.

- Le descripteur d'un arc (référence) reprend l'adresse du noeud pointé par cet arc et l'adresse du descripteur de la référence suivante.

- On peut accéder à la racine du graphe par un point d'entrée (entry point).

Après cet aperçu général, voyons ces différents descripteurs plus en détail.

A. Le dictionnaire.

A chaque noeud déclaré correspond une entrée dans le dictionnaire. Celle-ci contient le nom du noeud (8 bytes) ainsi que l'adresse du descripteur de ce noeud (4 bytes).

Les entrées sont triées par ordre alphabétique sur le nom des noeuds.

Les symboles BEGDIC et ENDIC, déclarés comme " entry point ", délimitent le début et la fin du dictionnaire.

B. Le descripteur d'une table.

La longueur de ce descripteur est de 28 bytes.

- Le premier élément est un code qui permet d'identifier le type de la table représentée. Ce code a une longueur de deux bytes et est déterminé par trois critères: l'existence d'une " dsect " qui décrit la table, l'unicité de la table, la manière dont sa longueur est donnée (par valeur ou par déplacement). Les différentes valeurs possibles sont indiquées dans le tableau ci-dessous.

Dsect	Y	Y	Y	Y	N	N	N	N
Unicité	Y	Y	N	N	Y	Y	N	N
Longueur	C	D	C	D	C	D	C	D
Code	1	2	3	4	5	6	7	8

- Le deuxième élément est le niveau de la table (2 bytes). Le graphe a, en effet, été découpé en niveaux; la racine est de niveau 1. Les autres tables ont un niveau qui représente la longueur du chemin le plus court entre elles et la racine.

- Le troisième élément est l'adresse de l'entrée correspondant à ce noeud dans le dictionnaire (4 bytes).

- Le quatrième élément est la longueur de la table (8 bytes).

On trouve ensuite les adresses des quatre types de références de la table.

- L'adresse (4 bytes) du premier élément de la liste des descripteurs des références simple qui ont pour origine cette table.

- L'adresse (4 bytes) du premier élément de la liste des descripteurs des références vecteur de la table.

- L'adresse (4 bytes) du premier élément de la liste des descripteurs des références multiple de la table.

- L'adresse (4 bytes) du premier élément de la liste des descripteurs des références retour. En effet, pour plus de commodité, il a été décidé de mentionner chaque fois l'adresse du ou des descripteurs des tables permettant d'accéder à celle-ci.

Remarque : Pour le dernier élément de chaque liste, l'adresse de l'élément suivant a une valeur égale à -1.

C. Le descripteur d'une référence simple.

La longueur de ce descripteur est de 20 bytes.

- Le code du descripteur en est le premier élément. Il prend la valeur 30 si un inverse de la référence simple existe, sinon il a la valeur 31.

- Le type du pointeur (2 bytes) est l'élément suivant. Si le pointeur est une adresse, son type est A suivi de la longueur du pointeur. Si le pointeur est une longueur de zone alors son type est L suivi de la longueur du pointeur.

- L'offset du pointeur (4 bytes) par rapport au début de la table est le troisième élément.

- Ensuite, on trouve l'adresse (4 bytes) du descripteur de la table contenant le pointeur.

- Le cinquième élément est l'adresse (4 bytes) du descripteur du noeud extrémité.

- L'adresse (4 bytes) du descripteur de la référence simple suivante, est le dernier élément.

D. Le descripteur d'une queue simple.

La longueur de ce descripteur est de 28 bytes.

- Le premier élément est un code (2 bytes) qui permet d'identifier le type de descripteur. Si la queue existe en un seul exemplaire, la valeur de ce code sera 10. Sinon, il vaudra 11.

- L'élément suivant est le niveau du noeud dans le graphe (2 bytes).

- L'adresse (4 bytes) de l'entrée du dictionnaire qui correspond à ce noeud, est le troisième élément.

- On trouve ensuite l'adresse (4 bytes) du descripteur du noeud constituant la queue.

- Pour l'alignement, il y a 2 bytes vides.

- Le sixième élément est le type du lien sur 2 bytes.

Viennent ensuite,

- L'offset (4 bytes) de ce lien.

- La valeur (4 bytes) de la fin de la queue.

- L'adresse (4 bytes) de la liste des descripteurs des références retour.

E. Le descripteur d'une queue double

Ce descripteur d'une longueur de 36 bytes est très semblable à celui d'une queue simple. Présentons rapidement son contenu:

- Un code (2 bytes) dont la valeur est 12 s'il y a unicité ou 13 dans le cas contraire.
- Le niveau (2 bytes).
- L'adresse (4 bytes) de l'entrée dans le dictionnaire.
- L'adresse (4 bytes) du descripteur du noeud composant la queue.
- Le type (2 bytes) du lien avant.
- Le type (2 bytes) du lien arrière.
- L'offset (4 bytes) du lien avant.
- L'offset (4 bytes) du lien arrière.
- La valeur (4 bytes) de fin de queue.
- L'adresse (4 bytes) du premier élément de la liste de pointeurs retour.

F. Le descripteur d'un anneau simple.

L'anneau simple a un descripteur d'une longueur de 24 bytes dont le contenu est :

- Un code (2 bytes) dont la valeur est 14 si l'anneau est unique ou 15 s'il existe en plusieurs exemplaires.

- Le niveau (2 bytes) de ce noeud.

- L'adresse (4 bytes) de l'entrée correspondante du dictionnaire.

- L'adresse (4 bytes) du descripteur du noeud composant l'anneau.

- 2 bytes d'alignement.

- Le type du lien (2 bytes).

- L'offset (4 bytes) de ce lien.

- L'adresse (4 bytes) du premier élément des références retour.

G. Le descripteur d'un anneau double.

Ce descripteur semblable à celui d'un anneau simple a une longueur de 32 bytes. Il contient:

- Le code sur 2 bytes (16 si unicité, sinon 17).
- Le niveau (2 bytes) du noeud dans le graphe.
- L'adresse (4 bytes) de l'entrée du dictionnaire.
- L'adresse (4 bytes) du descripteur du noeud composant l'anneau .
- Le type (4 bytes) du lien avant.
- Le type (4 bytes) du lien arrière.
- L'offset (4 bytes) du lien avant.
- L'offset (4 bytes) du lien arrière.
- L'adresse (4 bytes) du premier élément de la liste des références retour.

H. Le descripteur d'une référence vecteur.

La longueur de ce descripteur est de 24 bytes et les éléments le composant sont:

- Le code (2 bytes) dont la valeur est 40.

- Le type des composantes du vecteur (2 bytes).

- L'offset (4 bytes) de la première composante du vecteur, par rapport au début de la table.

- Le nombre (4 bytes) de composantes du vecteur.

- L'adresse (4 bytes) du descripteur de la table origine de l'arc.

- L'adresse (4 bytes) du descripteur de la table extrémité de l'arc.

- L'adresse (4 bytes) du premier descripteur de la liste des références retour.

I. Le descripteur d'une référence retour.

Ce descripteur de 16 bytes contient:

- Un code (2 bytes) qui permet d'identifier le type de référence dont ce descripteur est l'inverse. Les différentes valeurs possibles sont:
  - 60 pour une référence simple.
  - 61 pour une référence vecteur.
  - 62 pour une référence multiple.
  - 63 pour une queue simple.
  - 64 pour une queue double.
  - 65 pour un anneau simple.
  - 66 pour un anneau double.

- 2 bytes pour l'alignement.

- L'adresse (4 bytes) du descripteur du noeud contenant la référence.

- L'adresse (4 bytes) du descripteur du noeud extrémité.

- L'adresse (4 bytes) du descripteur du pointeur retour suivant.

J. La racine du graphe.

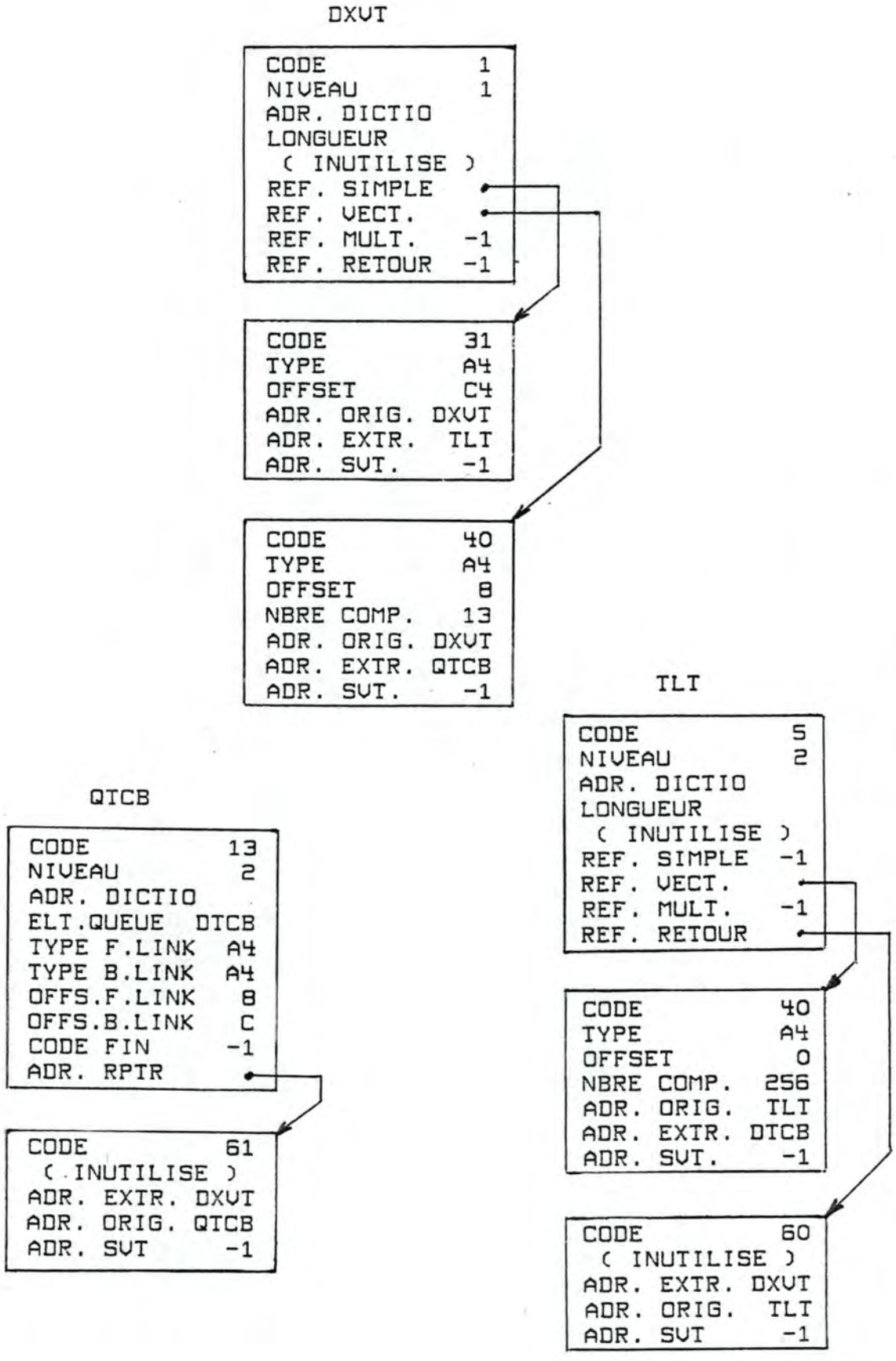
Le symbole ROOT déclaré comme entry point permet d'accéder à la racine du graphe.

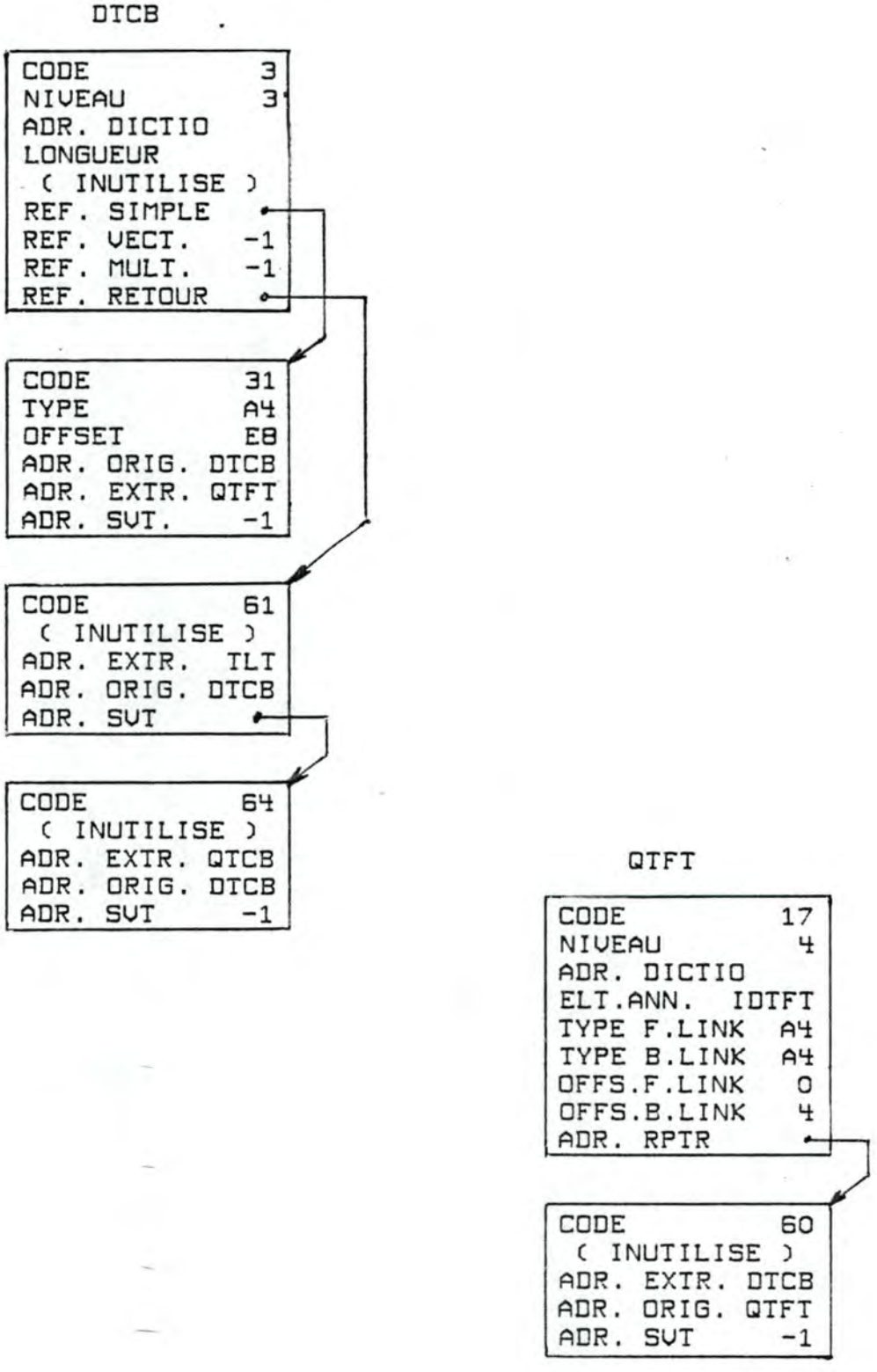
K. L'élément NIL

Ce noeud fictif d'une longueur de 4 bytes a la valeur 0.

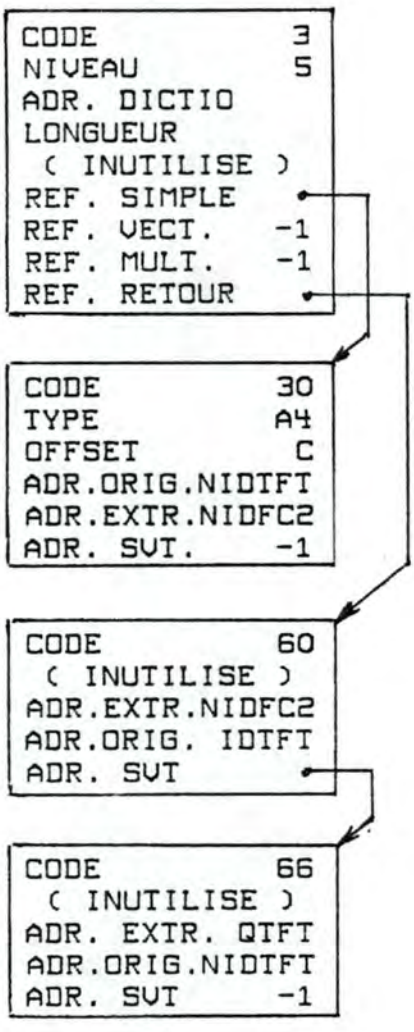
Remarque : Dans ce rappel de la représentation physique des descripteurs, nous n'avons pas mentionné le descripteur d'une partial page, ni celui d'une référence multiple. Ces deux descripteurs ne seront pas utilisés dans la suite de ce travail.

La figure suivante montre une schématisation de la représentation physique du graphe donné précédemment en exemple.

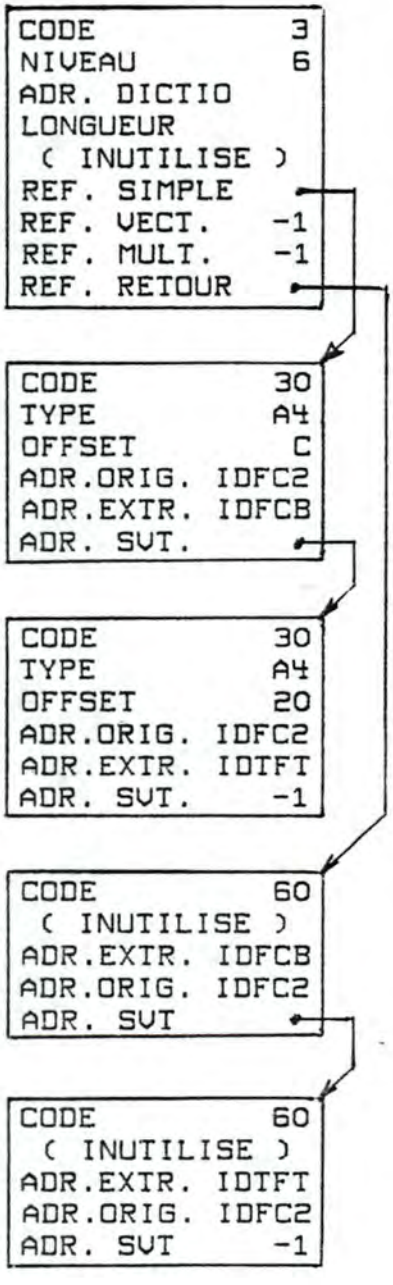

DXVT
CODE 1
NIVEAU 1
ADR. DICTIO
LONGUEUR
(INUTILISE)
REF. SIMPLE
REF. VECT.
REF. MULT. -1
REF. RETOUR -1
CODE 31
TYPE A4
OFFSET C4
ADR. ORIG. DXVT
ADR. EXTR. TLT
ADR. SVT. -1
CODE 40
TYPE A4
OFFSET 8
NBRE COMP. 13
ADR. ORIG. DXVT
ADR. EXTR. QICB
ADR. SVT. -1
TLT
CODE 5
NIVEAU 2
ADR. DICTIO
LONGUEUR
(INUTILISE)
REF. SIMPLE -1
REF. VECT.
REF. MULT. -1
REF. RETOUR
CODE 40
TYPE A4
OFFSET 0
NBRE COMP. 256
ADR. ORIG. TLT
ADR. EXTR. DICB
ADR. SVT. -1
CODE 60
(INUTILISE)
ADR. EXTR. DXVT
ADR. ORIG. TLT
ADR. SVT -1
QICB
CODE 13
NIVEAU 2
ADR. DICTIO
ELT.QUEUE DICB
TYPE F.LINK A4
TYPE B.LINK A4
OFFS.F.LINK 8
OFFS.B.LINK C
CODE FIN -1
ADR. RPTR
CODE 61
(INUTILISE)
ADR. EXTR. DXVT
ADR. ORIG. QICB
ADR. SVT -1


DTCB
CODE 3
NIVEAU 3
ADR. DICTIO
LONGUEUR
(INUTILISE)
REF. SIMPLE
REF. VECT. -1
REF. MULT. -1
REF. RETOUR
CODE 31
TYPE A4
OFFSET E8
ADR. ORIG. DTCB
ADR. EXTR. QTFT
ADR. SUT. -1
CODE 61
(INUTILISE)
ADR. EXTR. TLT
ADR. ORIG. DTCB
ADR. SUT
CODE 64
(INUTILISE)
ADR. EXTR. QTCB
ADR. ORIG. DTCB
ADR. SUT -1
QTFT
CODE 17
NIVEAU 4
ADR. DICTIO
ELT.ANN. IDTFT
TYPE F.LINK A4
TYPE B.LINK A4
OFFS.F.LINK 0
OFFS.B.LINK 4
ADR. RPTR
CODE 60
(INUTILISE)
ADR. EXTR. DTCB
ADR. ORIG. QTFT
ADR. SUT -1

IDTFT


CODE 3
NIVEAU 5
ADR. DICTIO
LONGUEUR
(INUTILISE)
REF. SIMPLE
REF. VECT. -1
REF. MULT. -1
REF. RETOUR
CODE 30
TYPE A4
OFFSET C
ADR.ORIG.NIDTFT
ADR.EXTR.NIDFC2
ADR. SVT. -1
CODE 60
(INUTILISE)
ADR.EXTR.NIDFC2
ADR.ORIG. IDTFT
ADR. SVT
CODE 66
(INUTILISE)
ADR. EXTR. QTFT
ADR.ORIG.NIDTFT
ADR. SVT -1

IDFC2


CODE 3
NIVEAU 6
ADR. DICTIO
LONGUEUR
(INUTILISE)
REF. SIMPLE
REF. VECT. -1
REF. MULT. -1
REF. RETOUR
CODE 30
TYPE A4
OFFSET C
ADR.ORIG. IDFC2
ADR.EXTR. IDFCB
ADR. SVT.
CODE 30
TYPE A4
OFFSET 20
ADR.ORIG. IDFC2
ADR.EXTR. IDTFT
ADR. SVT. -1
CODE 60
(INUTILISE)
ADR.EXTR. IDFCB
ADR.ORIG. IDFC2
ADR. SVT
CODE 60
(INUTILISE)
ADR.EXTR. IDTFT
ADR.ORIG. IDFC2
ADR. SVT -1

3.3 Les hypothèses et les spécifications du programme.

Pour permettre un fonctionnement correct du programme de parcours, il faut que la description du graphe soit conforme à la représentation physique définie par B. Bolle (Boll84) et dont nous venons de rappeler les principales caractéristiques.

Il faut également que la version de la structure de données, représentée par le graphe, corresponde à la version du dump pour que le programme de parcours soit capable de trouver la table désirée.

Nous avons décidé, de commun accord avec Mrs P. Dumont et B. Bolle , de ne pas implémenter le traitement des " partial page " ni celui des pointeurs multiples car ces structures ne seront pas employées, du moins dans un premier temps.

La fonction de ce programme de parcours est de donner à l'utilisateur l'adresse de la table qu'il souhaite atteindre, en lui permettant le choix du chemin à suivre pour aboutir à celle-ci. Cette fonction sera accomplie à condition que cette table soit décrite dans le graphe, qu'elle existe dans le dump et que les pointeurs menant à celle-ci, soient corrects.

3.4. Le programme de parcours d'un dump système.

3.4.1. L'analyse fonctionnelle.

La découpe de ce programme en applications, en phases et en fonctions nous paraît assez simple. Il forme un ensemble indépendant du reste du projet dans lequel nous pouvons distinguer deux parties,également indépendantes l'une de l'autre.

Nous pouvons donc considérer ce programme de parcours comme une application (Boda83). En effet, il constitue un traitement quasi autonome par rapport aux autres traitements de l'ensemble d'outils,défini précédemment.

Cette application est constituée de deux phases (Boda83):

- La recherche et le choix d'un chemin.
- Le parcours de ce chemin.

Ce sont bien deux phases car ces deux parties constituent des unités spatio-temporelles d'exécution. En effet, chacune s'exécute en un endroit et en une seule fois. Il n'y a pas de changement de ressources à l'intérieur de ces deux phases. Par contre, nous pourrions imaginer le stockage du chemin choisi afin de le parcourir ultérieurement.

Chacune de ces deux phases peut se décomposer en plusieurs fonctions (Boda83).

A. La recherche et le choix d'un chemin.

Le but de cette phase est de découvrir tous les chemins qui mènent à la table que l'utilisateur veut atteindre et de lui permettre de choisir celui qu'il désire emprunter pour accéder à cette table. La phase peut se décomposer en cinq fonctions principales.

- Saisie du nom de la table.

  Le programme demande à l'utilisateur le nom de la table qu'il souhaite atteindre.

- Vérification de l'existence de cette table.

  Le programme consulte le dictionnaire afin de vérifier si la table existe dans le graphe. Si celle-ci n'existe pas, un message d'erreur est affiché au terminal et le programme s'arrête. Sinon, le traitement continue.

- Recherche des chemins qui mènent à la table choisie.

  Le programme recherche tous les chemins qui, à partir de la racine du graphe, permettent d'accéder à la table désirée.

- Présentation de tous les chemins à l'utilisateur.

  Le programme affiche au terminal tous les chemins qu'il a trouvés et qui permettent d'accéder à la table souhaitée.

- Choix d'un chemin par l'utilisateur.

  Le programme demande à l'utilisateur de choisir un chemin parmi ceux qui lui sont proposés. Il vérifie également si la réponse de l'utilisateur correspond à un chemin proposé.

B. Le parcours de ce chemin.

Le but de cette phase est de parcourir le dump à l'aide de la description du chemin choisi par l'utilisateur et parvenir ainsi à trouver la table visée, afin de renvoyer à l'utilisateur l'adresse de cette table dans le dump.

Elle peut également se décomposer en plusieurs fonctions.

- Accès à la racine du graphe.

  Le programme cherche, dans le graphe, l'adresse de la racine de celui-ci.

- Accès à la racine du dump.

  Le programme cherche, dans le dump, l'adresse de la table correspondant à la racine du graphe c'est-à-dire, la racine du dump. Si le programme ne la trouve pas, il affiche un message d'erreur au terminal et s'arrête. Sinon, le programme continue le traitement.

- Parcours du chemin dans le graphe et dans le dump.

  Le programme accède au noeud suivant dans le graphe et cherche l'adresse de la table correspondante dans le dump jusqu'à la fin du chemin et donc, jusqu'à la table désirée. Si, dans le dump, le programme rencontre un pointeur dont la valeur n'est pas une adresse, il signale l'erreur en affichant un message au terminal et s'arrête.

3.4.2. La conception de l'architecture logicielle du programme.

Après l'analyse fonctionnelle du programme, il nous faut définir une architecture logicielle pour celui-ci. La première étape, dans la conception de cette architecture, est la structuration hiérarchique de ce logiciel.

A. La structure hiérarchique du système.

Nous allons essayer d'organiser le système suivant des niveaux distincts et ordonnés. Dans la mesure du possible, nous tenterons d'employer la hiérarchie " utilise " (Vlan84) pour réaliser cet ordonnancement.

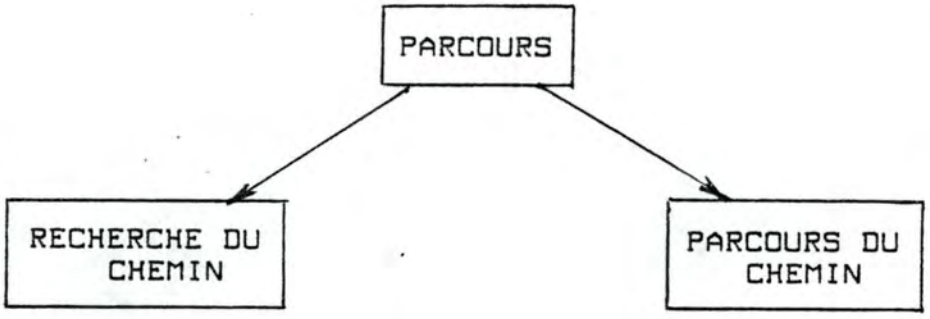
Rappelons brièvement les critères qui permettent de déterminer une relation " utilise ".

Soit deux composants, A et B, d'un système. Nous pourrons dire que A utilise B (Vlan84) si

- A est nettement plus simple du fait de pouvoir utiliser B.
- B n'est pas plus compliqué du fait de ne pas utiliser A.
- Il existe un sous-système " utile " qui contient B et qui ne contient pas A.
- Il n'existe pas de sous-système " utile " pouvant contenir A sans contenir B.

Un sous-système " utile " est un noyau du système capable de produire déjà des résultats.

Basons-nous sur le résultat de l'analyse fonctionnelle pour définir notre architecture logicielle. Nous avons une application que nous appellerons parcours, qui se divise en deux phases que nous nommerons respectivement recherche du chemin et parcours du chemin.



Remarque : Pour faciliter la compréhension, dans la suite, nous parlerons de noeuds pour le graphe, de tables pour le dump et d'étapes dans le chemin.

A1. La recherche du chemin.

Cette phase a pour objectif de rechercher les chemins qui mènent à la table que l'utilisateur souhaite atteindre, et de lui permettre de choisir celui qu'il veut parcourir.

La recherche du chemin peut se décomposer en plusieurs fonctions qui, à leur tour, peuvent se scinder en différents traitements.

a. La saisie du nom de la table.

Sortie : Le nom de la table que l'utilisateur désire atteindre.

Objectif : Demander à l'utilisateur le nom de la table dont il souhaite obtenir l'adresse.

b. Vérification de l'existence de cette table.

Entrée : Le nom de la table cherchée.

Sortie : L'adresse du noeud dans le graphe ou un message d'erreur.

Objectif : S'il existe dans le graphe, un noeud correspondant à la table, le programme renverra son adresse. Sinon, un message d'erreur est affiché au terminal.

Cette fonction utilise un autre élément.

b1. Adresse du noeud dans le graphe.

Entrée : Le nom du noeud.

Sortie : L'adresse du noeud dans le graphe ou -1.

Objectif : Consulter le dictionnaire du graphe. Si le nom du noeud s'y trouve, renvoyer l'adresse correspondante. Sinon, renvoyer -1.

c. Recherche de tous les chemins.

Entrée : L'adresse du noeud.

Sortie : Les chemins partant de la racine et aboutissant à ce noeud.

Objectif : En partant du noeud sélectionné, découvrir tous les chemins qui permettent de remonter jusqu'à la racine.

Cette fonction utilise plusieurs traitements.

c1. Niveau d'un noeud.

Entrée : L'adresse d'un noeud.

Sortie : Le niveau de ce noeud.

Objectif : Renvoyer le niveau du noeud dont l'adresse est donnée en entrée.

c2. Création d'une étape.

Entrée : L'adresse de l'étape courante.
L'adresse du noeud courant.
Le type du pointeur permettant d'accéder au noeud suivant.

Sortie : L'adresse de la nouvelle étape courante.

Objectif : Ajouter au chemin, une nouvelle étape comprenant l'adresse du noeud correspondant à cette étape (le noeud courant) et le type du pointeur qui permet d'accéder au noeud suivant.

c3. Recherche de tous les chemins qui partent du noeud courant.

Entrée : L'adresse du noeud courant.

Sortie : Les chemins partant de ce noeud vers la racine.

Objectif : Rechercher tous les chemins qui partent de ce noeud vers un autre noeud dont le niveau est inférieur ou égal à celui du noeud courant, en évitant les cycles du graphe.

Ce traitement s'effectue de manière récursive. On utilise, pour sa réalisation, les traitements suivants:

c3.1. Code du noeud.

Entrée : L'adresse d'un noeud.

Sortie : Le code de ce noeud.

Objectif : Renvoyer le code du noeud dont l'adresse est donnée en entrée.

c3.2. Adresse de la liste des pointeurs retour.

Entrée : L'adresse d'un noeud.

Sortie : L'adresse du premier pointeur retour.

Objectif : Renvoyer l'adresse du premier pointeur de la liste des pointeurs retour.

c3.3. Analyse du pointeur retour.

Entrée : L'adresse du descripteur d'un pointeur retour.

Sortie : L'adresse du pointeur retour suivant.
L'adresse du noeud pointé par le pointeur retour.
Le type du pointeur dont le pointeur retour est un inverse.

Objectif : Analyser le pointeur retour et renvoyer l'adresse du pointeur retour suivant, l'adresse du noeud pointé par le pointeur retour et le type du pointeur dont le pointeur retour est un inverse.

c3.4. Niveau du noeud.

Voir le point c1.

c3.5. Création d'une étape.

Voir le point c2.

c3.6. Création d'une racine.

Entrée : L'adresse du chemin courant.

Objectif : Ajouter à la chaîne des racines, une nouvelle racine qui pointe vers le chemin.

c3.7. Vérification des étapes mémorisées.

Entrée : L'adresse courante du chemin courant.
L'adresse du noeud courant.

Sortie : La valeur vrai ou faux.

Objectif : Renvoyer la valeur vrai si le noeud courant est déjà présent dans le chemin courant. Sinon,renvoyer la valeur faux.

d. Présentation de tous les chemins.

Entrée : Le pointeur vers la liste des racines des chemins.

Objectif : Afficher au terminal, tous les chemins qui ont été découverts.

Pour atteindre cet objectif, le traitement suivant est utilisé:

d1. Nom d'un noeud.

Entrée : L'adresse d'un noeud.

Sortie : Le nom de ce noeud.

Objectif : Renvoyer le nom du noeud dont l'adresse est donnée en entrée.

e. Choix d'un chemin.

Entrée : L'adresse de la liste des racines des chemins.

Sortie : L'adresse de la racine du chemin choisi par l'utilisateur.

Objectif : Permettre à l'utilisateur de choisir un chemin parmi ceux qui ont été découverts pour accéder à la table.

e1. Demande du choix.

Sortie : Le choix d'un chemin.

Objectif : Permettre à l'utilisateur de désigner le chemin qu'il désire emprunter pour accéder à la table.

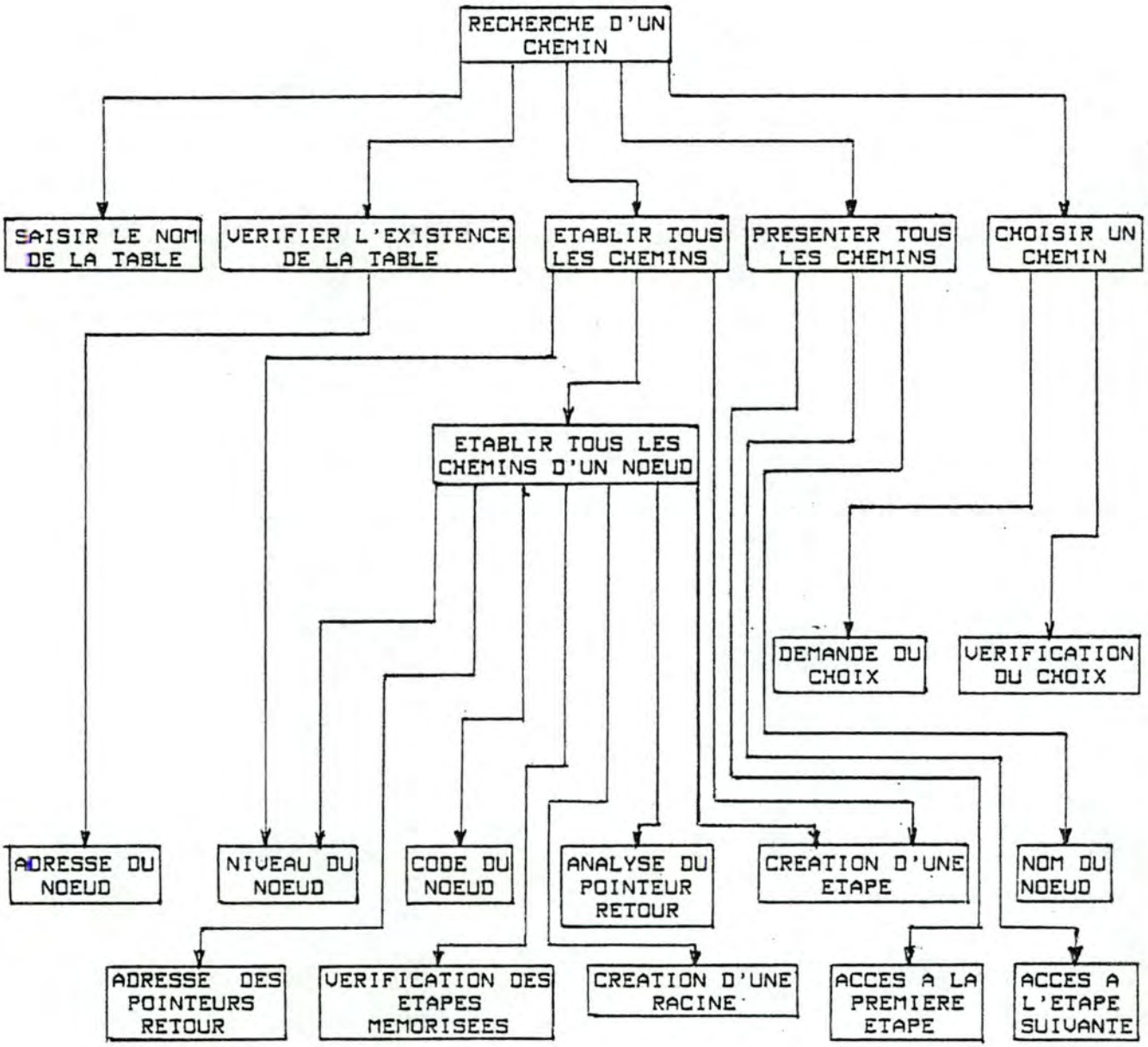
e2. Vérification du choix.

Entrée : Le choix de l'utilisateur.

Sortie : Un signal d'erreur si le choix ne convient pas.

Objectif : Vérifier si le chemin choisi par l'utilisateur est un chemin proposé.

Si nous représentons graphiquement cette hiérarchie, nous obtenons le schéma suivant :


RECHERCHE D'UN CHEMIN
SAISIR LE NOM DE LA TABLE
VERIFIER L'EXISTENCE DE LA TABLE
ETABLIR TOUS LES CHEMINS
PRESENTER TOUS LES CHEMINS
CHOISIR UN CHEMIN
ETABLIR TOUS LES CHEMINS D'UN NOEUD
DEMANDE DU CHOIX
VERIFICATION DU CHOIX
ADRESSE DU NOEUD
NIVEAU DU NOEUD
CODE DU NOEUD
ANALYSE DU POINTEUR RETOUR
CREATION D'UNE ETAPE
NOM DU NOEUD
ADRESSE DES POINTEURS RETOUR
VERIFICATION DES ETAPES MEMORISEES
CREATION D'UNE RACINE
ACCES A LA PREMIERE ETAPE
ACCES A L'ETAPE SUIVANTE

La phase de recherche du chemin se compose de 4 niveaux reliés entre eux par une relation " utilise ".

Niveau 1 : Outils d'accès au graphe et au chemin.

Au-dessus de ce niveau, la structure de représentation du graphe ainsi que celle du chemin, ne sont plus connues.

Niveau 2 : Choix d'un chemin et vérification de ce choix.

Au-dessus de ce niveau, la méthode de choix d'un chemin n'est plus connue.

Niveau 3 : Etablissement des chemins d'un noeud.

Au-dessus de ce niveau, la méthode de parcours du graphe est inconnue.

Niveau 4 : Fonctions définies lors de l'analyse fonctionnelle.

Les fonctions de ce niveau permettent la recherche d'un chemin.

A2. Le parcours du chemin.

Cette phase a pour objectif le parcours du chemin choisi par l'utilisateur. Pour chaque étape de ce chemin, il faut trouver la table correspondante dans le dump. Afin de réaliser cette recherche, plusieurs fonctions sont nécessaires.

a. Accès à la racine du graphe.

Entrée : L'adresse de la racine du chemin choisi.

Sortie : L'adresse de la racine du graphe.

Objectif : Obtenir l'adresse de la racine du graphe. Ce noeud correspond à la première étape du chemin.

Pour réaliser son objectif, cette fonction utilise les traitements suivants:

a1. Affichage du noeud atteint.

Entrée : L'adresse du noeud atteint.

Objectif : Afficher au terminal, le nom du noeud que le programme a atteint. C'est aussi le nom de la table que le programme va essayer de découvrir dans le dump.

a2. Nom d'un noeud.

Entrée : L'adresse d'un noeud.

Sortie : Le nom du noeud.

Objectif : Renvoyer le nom du noeud dont l'adresse est donnée en entrée.

b. Accès à la racine du dump.

Entrée : Le nom de la table correspondante à la racine du graphe.

Sortie : L'adresse de cette table dans le dump.

Objectif : Renvoyer l'adresse dans le dump de la table qui correspond à la racine du graphe.

Pour sa réalisation, cette fonction utilise les traitements suivants:

b1. Recherche d'une table dans le dump.

Entrée : Le nom de la table à chercher.

Sortie : L'adresse de cette table.

Objectif : Renvoyer l'adresse dans le dump de la table dont le nom est donné en entrée. Cet objectif ne peut être atteint que si le système connaît l'adresse de la table, ce qui est certainement le cas pour la racine.

b2. Vérification de la racine.

Entrée : L'adresse de la racine.

Sortie : Le code d'erreur.

Objectif : Si l'adresse de la racine n'est pas une adresse valide, renvoyer un code d'erreur correspondant au message à afficher. Sinon, renvoyer un code d'erreur avec une valeur égale à 0.

b3. Affichage d'un message d'erreur.

Entrée : Le code d'un message d'erreur.

Objectif : Afficher le message d'erreur correspondant au code donné en entrée.

c. Parcours du chemin dans le graphe et dans le dump.

Entrée : L'adresse de l'étape courante.
L'adresse de la table courante.
L'adresse du noeud courant.

Sortie : L'adresse de la table que l'utilisateur souhaite atteindre ou l'adresse de la dernière table courante.

Objectif : Parcourir le dump à l'aide du chemin et du graphe. S'il n'y a pas d'erreur, renvoyer l'adresse de la table que l'utilisateur souhaite atteindre. Sinon, afficher un message d'erreur et renvoyer l'adresse de la dernière table trouvée.

Cette fonction de parcours est assez complexe. Aussi, pour sa réalisation, elle utilise plusieurs traitements qui eux-mêmes sont constitués de plusieurs " sous-traitements ".

c1. Avance d'une étape et d'un noeud.

Entrée : Le type du lien entre le noeud courant et le noeud suivant.
L'adresse de l'étape courante.

Sortie : L'adresse de la nouvelle étape courante.
L'adresse du nouveau noeud courant.
L'adresse du descripteur du lien vers ce noeud.

Objectif : Chercher à l'aide du type de lien, celui qui mène du noeud courant vers le noeud suivant. Avancer à l'étape et au noeud suivant et renvoyer l'adresse du descripteur du pointeur ou de la queue, ou de l'anneau. Renvoyer également les adresses de la nouvelle étape et du nouveau noeud.

c1.1. Avance par pointeur.

Entrée : L'adresse du noeud origine.
L'adresse du noeud extrémité.

Sortie : L'adresse du descripteur du pointeur allant du noeud origine vers le noeud extrémité ou la valeur -1.

Objectif : S'il existe un pointeur entre le noeud origine et le noeud extrémité donnés en entrée, renvoyer l'adresse du descripteur du pointeur. Sinon, renvoyer la valeur -1.

c1.2. Avance par vecteur.

Entrée : L'adresse du noeud origine.
L'adresse du noeud extrémité.

Sortie : L'adresse du descripteur du vecteur allant du noeud origine vers le noeud extrémité ou la valeur -1.

Objectif : S'il existe un vecteur entre le noeud origine et le noeud extrémité donnés en entrée, renvoyer l'adresse du descripteur du vecteur. Sinon, renvoyer la valeur -1.

c1.3. Avance par queue.

Entrée : L'adresse du noeud origine.
L'adresse du noeud extrémité.

Sortie : L'adresse du descripteur de la queue allant du noeud origine vers le noeud extrémité ou la valeur -1.

Objectif : Si le noeud origine est une queue constituée du noeud extrémité, renvoyer l'adresse du descripteur de cette queue. Sinon, renvoyer la valeur -1.

c1.4. Avance par anneau.

Entrée : L'adresse du noeud origine.
L'adresse du noeud extrémité.

Sortie : L'adresse du descripteur de l'anneau allant du noeud origine vers le noeud extrémité ou la valeur -1.

Objectif : Si le noeud origine est un anneau constitué du noeud extrémité, renvoyer l'adresse du descripteur de cet anneau. Sinon, renvoyer la valeur -1.

c1.5. Affichage du noeud atteint.

Entrée : L'adresse d'un noeud.

Objectif : Afficher au terminal le nom du noeud que le programme a atteint.

c2. Avance dans le dump.

Entrée : Le type de lien entre la table courante et la suivante.
L'adresse du descripteur de ce lien.
L'adresse de la table courante.

Sortie : L'adresse de la nouvelle table courante.
Le code d'erreur.

Objectif : Chercher dans la table courante, le lien dont le descripteur est donné en entrée. Si ce lien existe, accéder à la table suivante et renvoyer son adresse ainsi qu'un code d'erreur de valeur 0. Sinon, renvoyer un code d'erreur dont la valeur correspond au message à afficher.

c2.1. Avance par pointeur.

Entrée : L'adresse de la table courante.
L'adresse du descripteur du pointeur.

Sortie : L'adresse de la nouvelle table courante.
Le code d'erreur.

Objectif : Appliquer à la table courante, un pointeur dont l'adresse du descripteur est donnée en entrée, pour trouver la table suivante. Si cette table est trouvée, renvoyer son adresse et un code d'erreur de valeur 0. Sinon, renvoyer un code d'erreur dont la valeur correspond au message à afficher.

c2.1.1. Données du pointeur.

Entrée : L'adresse du descripteur du pointeur.

Sortie : Le type du pointeur.
L'offset du pointeur.

Objectif : Renvoyer le type et l'offset du pointeur dont le descripteur est donné en entrée.

c2.1.2. Contenu de l'adresse.

Entrée : Une adresse dans le dump.

Sortie : Le contenu (une adresse) de cette adresse.

Objectif : Renvoyer le contenu du dump à l'adresse donnée en entrée.

c2.1.3. Vérification du contenu.

Entrée : Le contenu du dump obtenu en c2.1.2.
L'adresse courante dans le dump.

Sortie : L'adresse courante dans le dump.
Le code d'erreur.

Objectif : Si la valeur reçue en entrée n'est pas une adresse valide, renvoyer un code d'erreur correspondant au message à afficher et laisser inchangée, l'adresse courante dans le dump . Sinon, renvoyer un code dont la valeur est 0 et remplacer l'adresse courante par la valeur reçue.

c2.2. Avance par vecteur.

Entrée : L'adresse de la table courante.
L'adresse du descripteur du vecteur.

Sortie : L'adresse de la nouvelle table courante.
Le code d'erreur.

Objectif : Demander à l'utilisateur de choisir un des pointeurs composant le vecteur dont l'adresse du descripteur est donnée en entrée. Appliquer ce pointeur à la table courante, pour trouver la table suivante. Si cette table est trouvée, renvoyer son adresse et un code d'erreur de valeur 0. Sinon, renvoyer un code d'erreur dont la valeur correspond au message à afficher.

c2.2.1. Données du vecteur.

Entrée : L'adresse du descripteur du vecteur.

Sortie : Le type des pointeurs composant ce vecteur.
L'offset du vecteur.
Le nombre de composants du vecteur.

Objectif : Accéder au descripteur du vecteur dont l'adresse est donnée en entrée et renvoyer le type des pointeurs composant ce vecteur, l'offset du premier élément de ce vecteur, ainsi que le nombre d'éléments dans le vecteur.

c2.2.2. Choix du composant du vecteur.

Entrée : Le nombre de composants du vecteur.
L'offset du premier composant.

Sortie : L'adresse du composant choisi.

Objectif : Demander à l'utilisateur le composant du vecteur auquel il désire accéder. Vérifier le choix de l'utilisateur et renvoyer l'adresse de ce composant.

c2.2.3. Contenu de l'adresse.

Voir le point c2.1.2.

c2.2.4. Vérification du contenu.

Voir le point c2.1.3.

c2.3. Avance par queue.

Entrée : L'adresse de la table courante.
L'adresse du descripteur de la queue.

Sortie : L'adresse de la nouvelle table courante.
Le code d'erreur.

Objectif : Demander à l'utilisateur de sélectionner une table dans la queue dont l'adresse du descripteur est donnée en entrée. Si cette table est trouvée, renvoyer son adresse et un code d'erreur de valeur 0. Sinon, renvoyer un code d'erreur dont la valeur correspond au message à afficher.

c2.3.1. Données de la queue.

Entrée : L'adresse du descripteur de la queue.

Sortie : Le type du pointeur reliant les éléments de la queue.
L'offset de ce pointeur.

Objectif : Renvoyer le type et l'offset des pointeurs qui relient les composants de la queue.

c2.3.2. Choix d'une table.

Sortie : Le code du mode de sélection.
Les données relatives à ce mode de sélection.

Objectif : Demander à l'utilisateur de choisir un mode de sélection pour la table dans la queue. Ensuite, demander à l'utilisateur, les données relatives à ce mode choisi.

c2.3.3 Sélection d'une table.

Entrée : L'adresse de la table courante.
Le type des pointeurs reliant les éléments de la queue.
L'offset de ces pointeurs.
Le code du mode de sélection.
Les données de sélection.

Sortie : L'adresse de la table.
Le code d'erreur.

Objectif : Selon le mode de sélection, utiliser les données de sélection pour obtenir la table souhaitée. Si l'on parvient à trouver cette table, son adresse est renvoyée ainsi qu'un code d'erreur égal à 0. Sinon, le code d'erreur est renvoyé avec une valeur correspondante au message à afficher.

c2.3.3.1. Application d'un mode de sélection.

Entrée : Les données de sélection.
L'adresse de la table.

Sortie : Code de retour.

Objectif : Vérifier si la table courante est la table cherchée et dans ce cas, le code de retour a la valeur " trouvé ". Sinon, le code de retour a la valeur " pas trouvé ".

c2.3.3.2. Contenu de l'adresse.

Voir le point c2.1.2.

c2.3.4 Vérification d'erreur.

Entrée : Le code d'erreur.

Sortie : L'adresse de la table courante.

Objectif : S'il n'y a pas d'erreur, remplacer l'adresse de la table courante par l'adresse de la table sélectionnée dans la queue. Sinon, laisser l'adresse de la table courante, inchangée.

c2.4. Avance par anneau.

Entrée : L'adresse de la table courante.
L'adresse du descripteur de l'anneau.

Sortie : L'adresse de la nouvelle table courante.
Le code d'erreur.

Objectif : Demander à l'utilisateur de sélectionner une table dans l'anneau dont l'adresse du descripteur est donnée en entrée. Si cette table est trouvée, renvoyer son adresse et un code d'erreur de valeur 0. Sinon, renvoyer un code d'erreur dont la valeur correspond au message à afficher.

c2.4.1. Données de l'anneau.

Entrée : L'adresse du descripteur de l'anneau.

Sortie : Le type des pointeurs reliant les éléments de l'anneau.
L'offset de ces pointeurs.

Objectif : Renvoyerle type et l'offset des pointeurs qui relient les composants de l'anneau.

c2.4.2. Choix d'une table.

Voir le point c2.3.2

c2.4.3. Choix de la table.

Voir le point c2.3.3.

c2.4.4. Vérification d'erreur.

Voir le point c2.3.4.

d. Vérification du code d'erreur.

Entrée : Le code d'erreur.

Objectif : Si une erreur s'est produite, le code d'erreur n'a plus la valeur 0. Dans ce cas afficher un message d'erreur et s'arrêter.

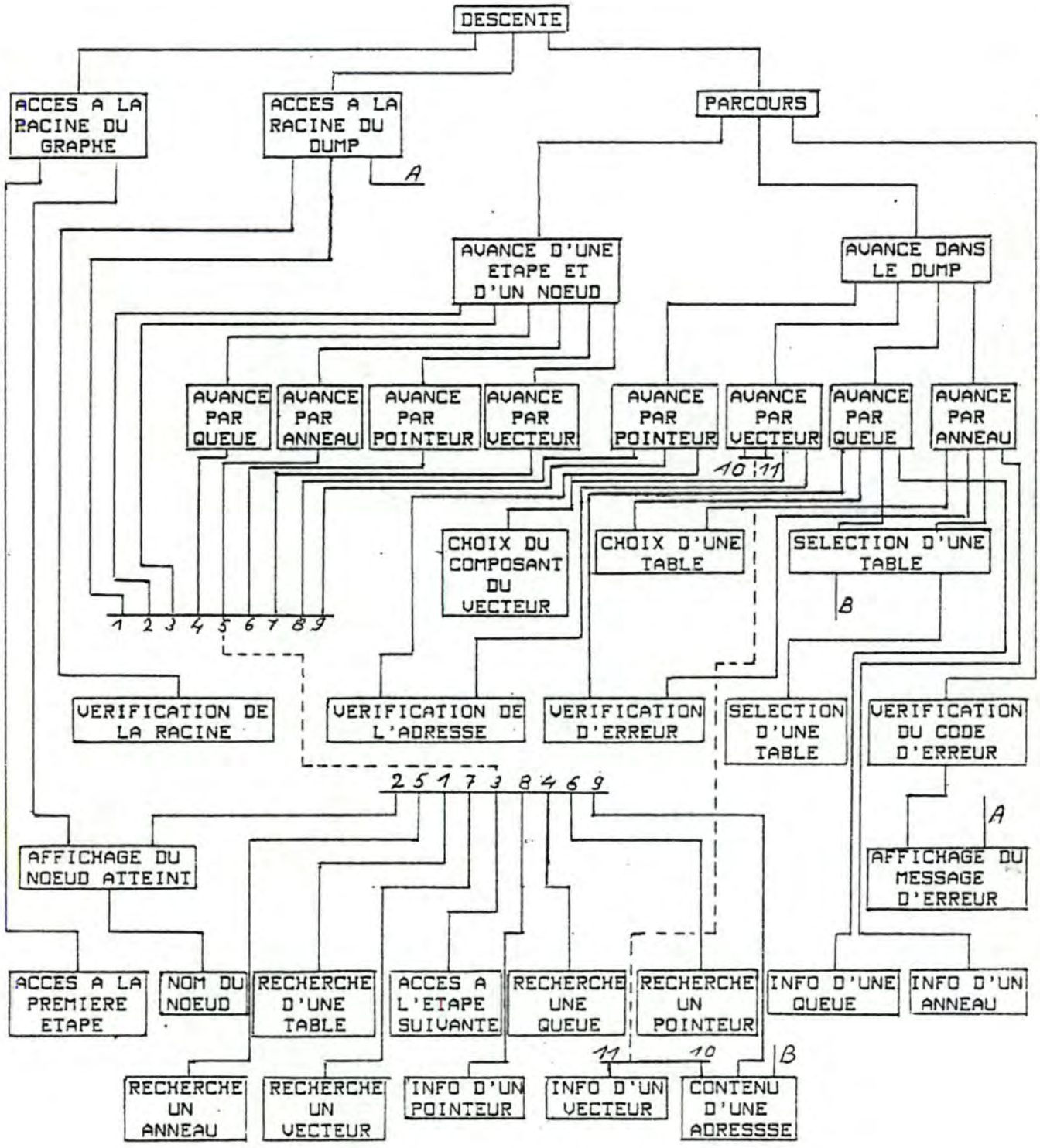
Cette fonction utilise le traitement suivant:

d1. Affichage d'un message d'erreur.

Entrée : Le code de l'erreur.

Objectif : Afficher au terminal, le message correspondant au code de l'erreur.

Si nous représentons graphiquement la décomposition de cette phase, nous obtenons le schéma ci-dessous.


DESCENTE
ACCES A LA RACINE DU GRAPHE
ACCES A LA RACINE DU DUMP
A
PARCOURS
AVANCE D'UNE ETAPE ET D'UN NOEUD
AVANCE DANS LE DUMP
AVANCE PAR QUEUE
AVANCE PAR ANNEAU
AVANCE PAR POINTEUR
AVANCE PAR VECTEUR
AVANCE PAR POINTEUR
AVANCE PAR VECTEUR
AVANCE PAR QUEUE
AVANCE PAR ANNEAU
10
11
CHOIX DU COMPOSANT DU VECTEUR
CHOIX D'UNE TABLE
SELECTION D'UNE TABLE
B
1 2 3 4 5 6 7 8 9
VERIFICATION DE LA RACINE
VERIFICATION DE L'ADRESSE
VERIFICATION D'ERREUR
SELECTION D'UNE TABLE
VERIFICATION DU CODE D'ERREUR
2 5 1 7 3 8 4 6 9
A
AFFICHAGE DU NOEUD ATTEINT
AFFICHAGE DU MESSAGE D'ERREUR
ACCES A LA PREMIERE ETAPE
NOM DU NOEUD
RECHERCHE D'UNE TABLE
ACCES A L'ETAPE SUIVANTE
RECHERCHE UNE QUEUE
RECHERCHE UN POINTEUR
INFO D'UNE QUEUE
INFO D'UN ANNEAU
11
10
B
RECHERCHE UN ANNEAU
RECHERCHE UN VECTEUR
INFO D'UN POINTEUR
INFO D'UN VECTEUR
CONTENU D'UNE ADRESSSE

La phase de parcours du chemin se compose de 7 niveaux reliés entre-eux par une relation " utilise ".

Niveau 1 : Outils d'accès au graphe, au chemin et au dump.

Au-dessus de ce niveau, la représentation physique du graphe, du chemin et du dump ne sont plus connues.

Niveau 2 : Affichage des messages à l'utilisateur.

Au-dessus de ce niveau, la gestion des informations destinées à l'utilisateur n'est plus connue.

Niveau 3 : Vérification des codes d'erreur et des critères de sélection.

Au-dessus de ce niveau, la notion d'erreur et les moyens de sélection des tables ne sont plus connus.

Niveau 4 : Choix et sélection.

Au-dessus de ce niveau, les composants d'un vecteur, d'une queue et d'un anneau ne sont plus connus.

Niveau 5 : Types de liens.

Au-dessus de ce niveau, l'implémentation des liens dans le graphe et dans le dump n'est plus connue.

Niveau 6 : Avance dans le chemin, le graphe et le dump.

Au-dessus de ce niveau, les éléments composants le chemin, le graphe et le dump ne sont plus connus.

Niveau 7 : Les fonctions de l'analyse fonctionnelle.

Les fonctions de ce niveau permettent le parcours d'un chemin.

Après la conception de la hiérarchie du logiciel, il faut que nous définissions une architecture modulaire (Vlan84).

B. Conception d'une structure modulaire.

Afin d'obtenir une structure modulaire, nous devons reprendre les différents traitements établis lors de la conception de la hiérarchie du logiciel et les regrouper en modules.

Nous utilisons trois types de structures différents : le graphe, le chemin et le dump. Nous pouvons donc regrouper les fonctions d'accès à ces structures afin d'établir trois modules d'interface.

- Le module " interface avec le graphe ".

Il permet les opérations suivantes :

- Obtenir l'adresse d'un noeud à partir de son nom.
- Obtenir le nom d'un noeud à partir de son adresse.
- Obtenir le niveau d'un noeud à partir de son adresse.
- Obtenir le code d'un noeud à partir de son adresse.
- Obtenir l'adresse de la liste des pointeurs retour du noeud à partir de son adresse.
- Analyser un pointeur retour.
- Chercher un pointeur.
- Chercher un vecteur.
- Chercher une queue.
- Chercher un anneau.
- Obtenir les informations d'un pointeur.
- Obtenir les informations d'un vecteur.
- Obtenir les informations d'une queue.
- Obtenir les informations d'un anneau.

Remarque : Les spécifications de ces opérations sont décrites dans la partie concernant la conception d'une structure hiérarchique.

- Le module " interface avec le chemin ".

Ce module regroupe les opérations suivantes :

- Création d'une racine.
- Création d'une étape.
- Accès à la première étape.
- Accès à l'étape suivante.
- Vérification de la mémorisation d'une étape dans le chemin courant.

- Le module " interface avec le dump ".

Ce module regroupe deux opérations :

- Accès à la racine.
- Obtenir le contenu d'une adresse.

En plus de ces interfaces, nous avons regroupé les autres traitements en trois modules.

- Le module " recherche d'un chemin ".

Ce module permet les traitements suivants:

- Saisie du nom du noeud à atteindre.
- Vérification de l'existence de ce noeud.
- Recherche de tous les chemins vers ce noeud.
- Présentation de tous les chemins découverts.
- Choix d'un chemin.

- Le module " parcours du chemin ".

Ce module reprend les opérations suivantes :

- Accès à la racine du graphe.
- Accès à la racine du dump.
- Parcours du chemin.

- Le module " Affichage des messages d'erreurs ".

Ce module assure la gestion et l'affichage des messages d'erreurs.

Un module séquenceur a également été implémenté afin de permettre un ordonnancement des opérations entre les différents modules.

3.4.3. Quelques détails de l'implémentation.

A. Le choix des langages de programmation.

L'interface du graphe et l'interface du dump ont été réalisés en assembleur.

L'interface du dump utilise les macro instructions d'un outil d'accès aux dumps (A.I.D.).

Les autres modules ont été réalisés en Pascal car ce langage, qui permet la récursivité, était directement disponible.

B. Choix de représentation des chemins.

Nous avons choisi, pour la représentation physique des chemins, une structure de " records " Pascal créés dynamiquement.

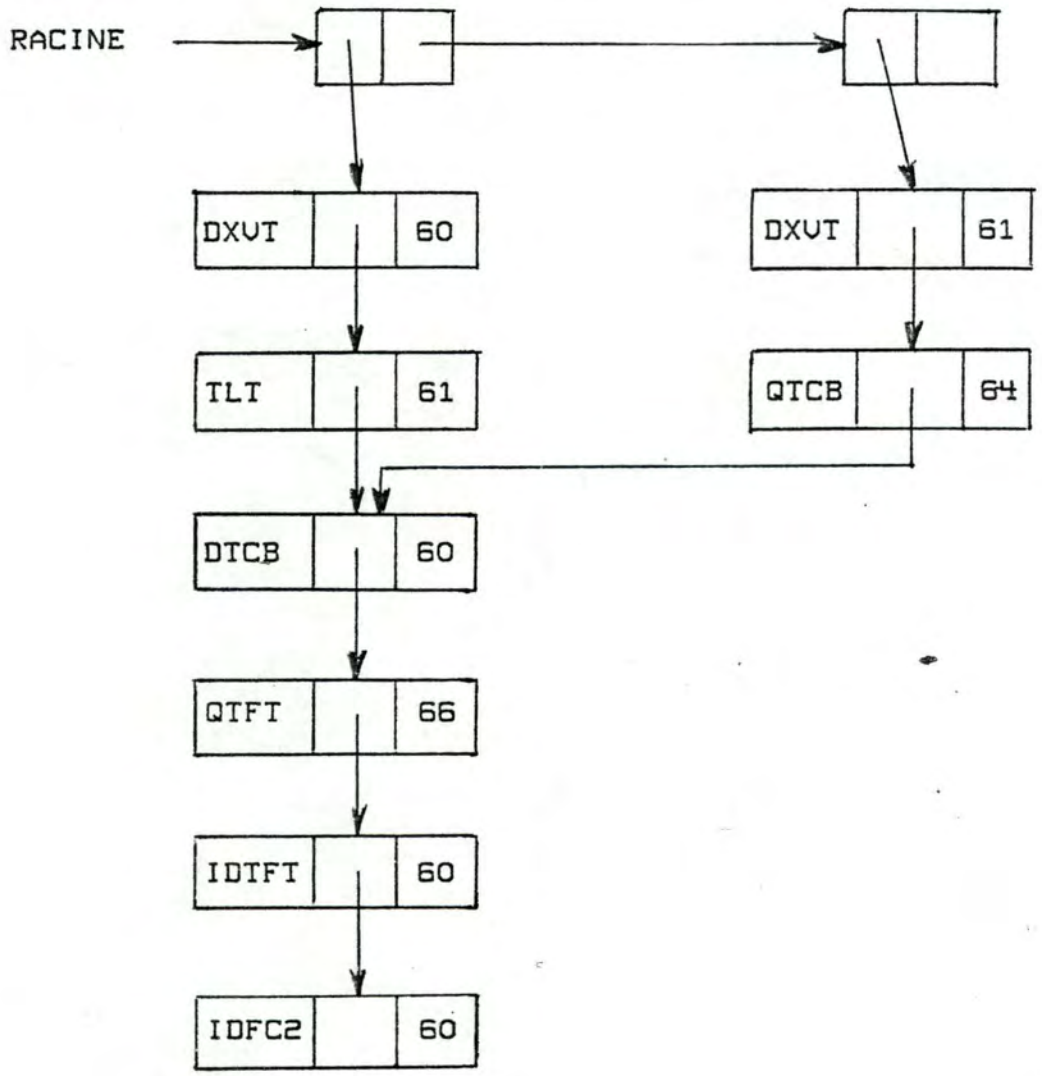
Chaque " record " représentant une étape, possède trois champs:

- un pointeur vers l'étape suivante.
- un pointeur vers le noeud correspondant.
- le type du lien entre le noeud correspondant à cette étape et le noeud suivant.

Ces " records " sont chaînés et constituent un chemin. Celui-ci possède une racine qui est représentée par un autre " record " à deux champs :

- un pointeur vers la racine suivante.
- un pointeur vers la première étape du chemin.

Les racines sont également liées les unes aux autres et l'adresse de la première est contenue dans une variable, ce qui permet l'accès aux racines et donc aux chemins. Nous pouvons représenter cette structure par le schéma suivant :



De cette manière, il est possible de parcourir un chemin et de retrouver rapidement le noeud qui correspond à chaque étape puisque l'adresse de son descripteur est contenue dans celle-ci.

C. Méthode de qualification des tables.

Certaines tables, telle les TCB, existent en plusieurs exemplaires dans le dump. Il faut donc, lors du parcours de ce dump, permettre à l'utilisateur de choisir une de ces tables. Nous avons opté pour un choix, au fur et à mesure des besoins , plutôt que pour une qualification complète du chemin avant de commencer le parcours. C'est-à-dire que, chaque fois que le programme rencontre un " distributeur " (un vecteur, une queue ou un anneau), il demande à l'utilisateur de choisir une composante du vecteur ou une table de la queue ou de l'anneau.

D. La recherche des chemins.

Nous avons décidé d'implémenter de manière récursive la recherche de tous les chemins partant d'un noeud et aboutissant à la racine du graphe. En effet, cette recherche s'effectue par le parcours d'un graphe en appliquant la méthode " en profondeur d'abord ".

E. Le " link " et le chargement du programme.

Le programme est d'abord " linker " avec le graphe, les deux interfaces d'accès au graphe et au dump, et également avec le fichier système " Pascalrt ". Ensuite, il est chargé par la commande " load " et exécuté en tapant R.

4. CONCLUSION

4.1 Résultat.

Toutes les fonctions des interfaces ont été testées pour tous les cas possibles. Elles remplissent correctement leur rôle.

Pour le test du programme, nous disposions d'un exemple réduit de graphe qui contenait cependant toutes les structures définies au point 3.2. Les différents tests effectués n'ont pas révélés d'erreur dans le programme.

Ce programme permet donc de retrouver l'adresse d'une table dans un dump. Mais comme nous l'avons dit auparavant, il ne constitue qu'une partie de l'ensemble d'outils dont la plupart reste à réaliser.

4.2. Possibilités d'amélioration.

Certaines modifications peuvent être envisagées afin d'améliorer le programme. Citons-en quelques-unes.

4.2.1. Le choix de tables.

La sélection de tables multiples dans une queue ou un anneau est effectuée, pour l'instant, en donnant la position de la table souhaitée dans cette queue ou cet anneau. Nous pourrions envisager de permettre à l'utilisateur de demander au programme la sélection de la table qui a une valeur donnée à un déplacement donné également.

Ou encore, nous pourrions employer les " dsects " des tables afin que l'utilisateur précise la valeur d'un élément de la table qu'il veut choisir.

4.2.2. La qualification du chemin.

Nous l'avons dit plus haut, la qualification du chemin se fait au fur et à mesure de la nécessité d'un choix. Mais, nous

pourrions demander à l'utilisateur de qualifier complètement le chemin avant de commencer le parcours.

4.2.3. Le stockage du chemin choisi.

Actuellement, il n'y a pas de mémorisation du chemin choisi par l'utilisateur. Nous pourrions imaginer de permettre à l'utilisateur le stockage du chemin qu'il désire emprunter. Cela lui donnerait la possibilité de parcourir plusieurs dumps de la même manière sans devoir, à chaque fois, rechercher tous les chemins qui, dans le graphe, mènent à la table voulue.

4.2.4. La gestion des écrans.

Dans le programme actuel, la gestion des écrans est réduite au minimum. Une gestion plus élaborée pourrait-être envisagée afin de rendre l'utilisation du programme plus agréable.

Bibliographie.

(Boda83) Bodart F. , Cours d'analyse fonctionnelle, Institut d'informatique, Namur, 1983.

(Boll84) Bolle B. , " Représentation formelle de structures de données. Application au système BS2000 ", Mémoire, Institut d'informatique, Namur, 1984-85.

(Damp) " Benutzerbeshreibung V1.1. ", Siemens.

(Deco82) Decoene H, " Helga V624. Measurement and Debugging programs ", Siemens Data Brussels, Bruxelles, Mai 82.

(Dumo85) Dumont P. , " Introduction aux dumps ", Siemens Software Namur, Namur, Janvier 85.

(Grem84) Gremillion L. , " Determinants of program repair maintenance requirement ", Communications of the ACM, Vol 27, N 8, Août 84, Pp 826-832.

(Hirs86) Hirschvogel, " AID ", Siemens, Avril 86.

(Kais83) Kaiser , " External-Interface AID ", Siemens Datenverarbeitung, Muenchen, Novembre 83.

(Pari86) Parikh G. , " Exploring the world of software maintenance. I: What is software maintenance ? ", ACM Software engineering notes, Vol 11, N 2, Avril 86, Pp 49-52.

(Vlan84) Van Lansweerde A. , Cours de méthodologie de développement de logiciels, Institut d'informatique, Namur, 1984.

(Zelk78) Zelkowitz M., " Perspectives on software engineering ", Computing Surveys, Vol 10, N 2, Juin 78, Pp 197-216.

ANNEE ACADEMIQUE 1985-86.

MAINTENANCE DU SYSTEME D'EXPLOITATION BS 2000.

OUTIL DE PARCOURS DE DUMPS.

ANNEXE

MEMOIRE PRESENTE PAR
MARCHAL PATRICK EN VUE
DE L'OBTENTION DU GRADE
DE LICENCIE ET MAITRE
EN INFORMATIQUE.

PROMOTEURS :

J. RAMAEKERS
P. DUMONT.

TABLE DES MATIERES.
==========

```
 1  1  0    (*$Y+,U+,W-*)
 2  1  0    PROGRAM PARCOURS (INPUT,OUTPUT);
 3  1  0
 4  1  0    CONST
 5  1  0
 6  1  0      NUL = -1;                                   (* ADRESSE NULLE *)
 7  1  0      BL8 = '        ';                           (* 8 BLANCS *)
 8  1  0
 9  1  0    TYPE
10  1  0      ADRESSE = INTEGER;
11  1  0      CH2     = PACKED ARRAY [1..2] OF CHAR;
12  1  0      CH8     = PACKED ARRAY [1..8] OF CHAR;
13  1  0      ETAPTR  = ^ETAPE;
14  1  0      RACPTR  = ^RACIN;
15  1  0      TYPEL   = INTEGER;
16  1  0
17  1  0      ETAPE   = RECORD
18  1  1+                 DESCADR : ADRESSE;(* ADRESSE DU NOEUD CORRESPONDANT*)
19  1  1                  PSUIV   : ETAPTR; (* PTR ETAPE SUIVANTE           *)
20  1  1                  TYPELIEN: TYPEL;  (* TYPE DU PTR VERS NOEUD SVT    *)
21  1  1                END;
22  1  0-
23  1  0      RACIN   = RECORD
24  1  1+                 PETAPE  : ETAPTR; (* PTR VERS ETAPE *)
25  1  1                  PSUIV   : RACPTR; (* PTR VERS RACINE SUIVANTE*)
26  1  1                END;
27  1  0-
28  1  0    VAR
29  1  0      CURRAC   : RACPTR;   (* PTR VERS RACINE COURANTE (VAR GLOBALE)  *)
30  1  0      RACINE   : RACPTR;   (* PTR VERS RACINE CHEMIN CHOISI PAR L'UTILIS *)
31  1  0
32  1  0
33  1  0
34  1  0
35  1  0    (*=========================================================================*)
36  1  0
37  1  0    PROCEDURE ERREUR ( CODE : INTEGER );
38  2  0
39  2  0    (*
40  2  0 C    PRECOND  : < CODE > EST LE CODE DE L'ERREUR QUE L'ON VEUT SIGNALER
41  2  0 C
42  2  0 C    FONCTION : SIGNALE A L'UTILISATEUR L'OCCURENCE D'UNE ERREUR
43  2  0 C
44  2  0 C    POSTCOND :
45  2  0 C *)
46  2  0
47  2  0      BEGIN
48  2  1+      WRITELN;
49  2  1       WRITELN ('                              **************************');
50  2  1       WRITELN ('                              *    E R R E U R   *');
51  2  1       WRITELN ('                              **************************');
52  2  1       WRITELN;
53  2  1       WRITELN;
54  2  1       CASE CODE OF
55  2  2+        1 : WRITELN ('RACINE NON TROUVEE OU FICHIER N''EXISTE PAS ');
56  2  2         2 : WRITELN ('POINTEUR VERS LE NOEUD SUIVANT NON TROUVE ');
57  2  2         3 : WRITELN ('TABLE NON TROUVEE DANS LE DUMP ');
58  2  2       END;  (* CASE *)
59  2  1-    END ;

END ERREUR   L=   444
                  (* ERREUR *)
60  1  0-
61  1  0   (*=========================================================================*)
```

```
  62  1  0
  63  1  0
  64  1  0   PROCEDURE CREERRACINE (CHEMIN : ETAPTR);
  65  2  0
  66  2  0   (*
  67  2  0 C   ENTREE    : CHEMIN  : ADRESSE DU CHEMIN
  68  2  0 C
  69  2  0 C   FONCTION : AJOUTE A LA CHAINE DES RACINES (VAR GLOBALE) UNE NOUVELLE
  70  2  0 C                RACINE VERS LE CHEMIN
  71  2  0 C *)
  72  2  0
  73  2  0   VAR
  74  2  0     RAC : RACPTR;
  75  2  0
  76  2  0   BEGIN
  77  2  1+    WRITELN ('                          DEBUT CREATION RACINE ');
  78  2  1     NEW (RAC);
  79  2  1     CURRAC^.PSUIV := RAC;
  80  2  1     RAC^.PETAPE := CHEMIN;
  81  2  1     RAC^.PSUIV := NIL;
  82  2  1     CURRAC := RAC;
  83  2  1     WRITELN ('                          FIN CREATION RACINE ');
  84  2  1   END;

END CREERRAC L=   170

  85  1  0-
  86  1  0   (*=================================================================*)
  87  1  0
  88  1  0   PROCEDURE CREERETAPE (VAR CH : ETAPTR; ADR : ADRESSE; TYP : TYPEL);
  89  2  0
  90  2  0   (*
  91  2  0 C   PRECOND  : <CH> POINTE VERS L'ETAPE COURANTE DU CHEMIN
  92  2  0 C              <ADR> ADR DU NOEUD CORRESPONDANT A L'ETAPE A AJOUTER
  93  2  0 C              <TYP> TYPE DU PTR PERMETTANT D'ACCEDER AU NOEUD SUIVANT
  94  2  0 C
  95  2  0 C   FONCTION : AJOUTE AU CHEMIN UNE NOUVELLE ETAPE APRES L'ETAPE COURANTE
  96  2  0 C
  97  2  0 C   POSTCOND : <CH> POINTE VERS LA NOUVELLE ETAPE COURANTE
  98  2  0 C
  99  2  0 C *)
 100  2  0
 101  2  0   VAR
 102  2  0     PTR  : ETAPTR;
 103  2  0
 104  2  0   BEGIN
 105  2  1+    WRITELN ('                            DEBUT CREATION ETAPE ');
 106  2  1     NEW (PTR);
 107  2  1     PTR^.DESCADR := ADR;
 108  2  1     PTR^.PSUIV := CH;
 109  2  1     PTR^.TYPELIEN := TYP;
 110  2  1     CH := PTR;
 111  2  1     WRITELN ('                            FIN CREATION ETAPE ');
 112  2  1   END;

END CREERETA L=   180

 113  1  0-
 114  1  0   (*=================================================================*)
```

```
 115  1  0
 116  1  0   FUNCTION MEMORISE(CHEMIN:ETAPTR;DESCADR:ADRESSE):BOOLEAN;
 117  2  0
 118  2  0   (*
 119  2  0 C   ENTREE   : CHEMIN  : PTR VERS CHEMIN DEJA MEMORISE
 120  2  0 C              DESCADR : ADRESSE DU NOEUD DONT LA MEMORISATION EST A
 121  2  0 C                        VERIFIER
 122  2  0 C
 123  2  0 C   FONCTION : VERIFIE SI LE NOEUD DONT L'ADRESSE EST < DESCADR > A DEJA
 124  2  0 C              ETE MEMORISEE DANS LE < CHEMIN > .
 125  2  0 C
 126  2  0 C   SORTIE   : MEMORISE A LA VALEUR VRAI SI LE NOEUD A DEJA ETE MEMORISEE
 127  2  0 C                          FAUX SI LE NOEUD N'A PAS ENCORE ETE MEMORISEE
 128  2  0 C
 129  2  0 C *)
 130  2  0
 131  2  0   VAR
 132  2  0      TROUVE : BOOLEAN;          (* VRAI SI ON A TROUVE LE NOEUD *)
 133  2  0      PTR    : ETAPTR;           (* PTR VERS ETAPE COURANTE *)
 134  2  0
 135  2  0   BEGIN
 136  2  1+     WRITELN ('        DEBUT MEMORISE ');
 137  2  1      TROUVE := FALSE;
 138  2  1      PTR := CHEMIN;
 139  2  1      WHILE (PTR <> NIL) AND (NOT TROUVE) DO BEGIN
 140  2  2+       IF PTR^.DESCADR = DESCADR THEN TROUVE := TRUE
 141  2  2        ELSE PTR := PTR^.PSUIV;
 142  2  2      END;
 143  2  1-     IF TROUVE THEN MEMORISE := TRUE
 144  2  1      ELSE MEMORISE := FALSE;
 145  2  1      WRITELN ('               FIN MEMORISE ');
 146  2  1   END;

END MEMORISE L=   232

 147  1  0-
 148  1  0   (*=====================================================================*)
 149  1  0
 150  1  0   PROCEDURE NODEDIC (VAR NOM :CH8; VAR ADR :ADRESSE);EXTERNAL;
 151  1  0
 152  1  0   (*
 153  1  0 C   PRECOND  :
 154  1  0 C
 155  1  0 C   FONCTION : RENVOI DANS < ADR > L'ADRESSE DU NOEUD DONT LE NOM EST
 156  1  0 C              < NOM > S'IL EXISTE. SINON < ADR > VAUT -1
 157  1  0 C
 158  1  0 C   POSTCOND : < ADR > = L'ADRESSE DE NOM S'IL EXISTE, SINON <ADR> = -1
 159  1  0 C *)
 160  1  0
 161  1  0   (*=====================================================================*)
```

```
162  1  0
163  1  0   PROCEDURE CODENODE (DESCADR : ADRESSE; VAR CODE : INTEGER); EXTERNAL;
164  1  0
165  1  0   (*
166  1  0 C   PRECOND  : <DESCADR> = ADR VALIDE DU NOEUD DONT ON VEUT CONNAITRE LE
167  1  0 C                          CODE
168  1  0 C
169  1  0 C   FONCTION : RENVOI DANS < CODE > LE CODE DU NOEUD DONT L' ADR EST
170  1  0 C              < DESCADR >
171  1  0 C
172  1  0 C   POSTCOND : < CODE > = LE CODE DU NOEUD
173  1  0 C *)
174  1  0
175  1  0   (*================================================================*)
176  1  0
177  1  0   PROCEDURE ADRREPTR (DESCR : ADRESSE; VAR RPTR : ADRESSE;
178  2  0                       CODE : INTEGER); EXTERNAL;
179  1  0
180  1  0   (*
181  1  0 C   PRECOND  : < DESCR > EST L'ADRESSE VALIDE D'UN NOEUD ET < CODE > EST
182  1  0 C              LE CODE DE CE NOEUD.
183  1  0 C
184  1  0 C   FONCTION : RENVOI DS < RPTR > L'ADRESSE DU PREMIER POINTEUR RETOUR
185  1  0 C              S'IL EXISTE. SINON < RPTR > VAUT  -1
186  1  0 C
187  1  0 C   POSTCOND :  < RPTR > = ADRESSE DU PREMIER PTR RETOUR S'IL EXISTE
188  1  0 C               SINON < RPTR > = -1
189  1  0 C *)
190  1  0
191  1  0   (*================================================================*)
192  1  0
193  1  0   PROCEDURE ANALRPTR (VAR PTR : ADRESSE; VAR SVT : ADRESSE; VAR TYP:TYPEL;
194  2  0                       VAR NODE : ADRESSE);EXTERNAL;
195  1  0
196  1  0   (*
197  1  0 C   PRECOND  : < PTR > = ADRESSE DU POINTEUR RETOUR A ANALYSER
198  1  0 C
199  1  0 C   FONCTION : RENVOI DS < SVT > L'ADRESSE DU PTR RETOUR SUIVANT (-1 S'IL
200  1  0 C              N'EXISTE PAS), DS < NOEUD > L'ADRESSE DU DESCR (D1) POINTE
201  1  0 C              PAR LE PTR RETOUR ET DS <TYPE >, LE TYPE DU PTR PERMETTANT
202  1  0 C              D'ACCEDER A CE DESCR(D2) A PARTIR DU DESCR PRECEDANT (D1).
203  1  0 C
204  1  0 C   POSTCOND : <SVT>= ADRESSE DU RPTR SUIVANT S'IL EXISTE,SINON <SVT>= -1
205  1  0 C              <NOEUD>= ADRESSE DU DESCRIPTEUR CORRESPONDANT
206  1  0 C              <TYPE>= TYPE DU PTR VERS CE DESCRIPTEUR
207  1  0 C *)
208  1  0

210  1  0
211  1  0   PROCEDURE NIVNOEUD (VAR DESCADR : ADRESSE; VAR NIVEAU : INTEGER);
212  2  0                       EXTERNAL;
213  1  0
214  1  0   (*
215  1  0 C   PRECOND  : < DESCADR> = ADR VALIDE DU NOEUD DONT ON VEUT CONNAITRE LE
216  1  0 C                          NIVEAU
217  1  0 C
218  1  0 C   FONCTION : RENVOI DANS < NIVEAU > LE NIVEAU DU NOEUD
219  1  0 C
220  1  0 C   POSCOND  : < NIVEAU > = NIVEAU DU NOEUD
221  1  0 C *)
222  1  0
223  1  0   (*================================================================*)
224  1  0
225  1  0   PROCEDURE NAMENODE (VAR ADRESS : ADRESSE ; VAR NAME : CH8);EXTERNAL;
226  1  0
227  1  0   (*
228  1  0 C   PRECOND  : <ADRESS> EST L'ADRESSE VALIDE D'UN NOEUD
229  1  0 C
230  1  0 C   FONCTION : RENVOI DANS <NAME> LE NOM DU NOEUD DONT L'ADR EST <ADRESS>
231  1  0 C
232  1  0 C   POSTCOND : <NAME> CORRESPOND AU NOM DU NOEUD DONT L'ADR EST <ADRESS>
233  1  0 C *)
234  1  0
235  1  0   (*================================================================*)
```

```
  236  1  0
  237  1  0   PROCEDURE ETABLICHEMIN ( CHEMIN : ETAPTR;  NIVEAU : INTEGER);
  238  2  0
  239  2  0   (*
  240  2  0 C   ENTREE   : CHEMIN   : PTR VERS LE CHEMIN DEJA CONNU
  241  2  0 C              NIVEAU   : NIVEAU DE LA DERNIERE ETAPE DU CHEMIN
  242  2  0 C
  243  2  0 C   FONCTION : AJOUTE AU CHEMIN UNE NOUVELLE ETAPE DONT LE NIVEAU EST <
  244  2  0 C              AU NIVEAU DE LA DERNIERE ETAPE OU DONT LE NIVEAU EST =
  245  2  0 C              MAIS QUI N'A PAS ENCORE ETE MEMORISEE
  246  2  0 C
  247  2  0 C *)
  248  2  0
  249  2  0   VAR
  250  2  0
  251  2  0     CODE    : INTEGER;  (* CODE DU NOEUD COURANT *)
  252  2  0     CURCH   : ETAPTR;   (* PTR VERS CHEMIN COURANT *)
  253  2  0     DESCADR : ADRESSE;  (* ADRESSE DU NOEUD COURANT *)
  254  2  0     FIN     : BOOLEAN;
  255  2  0     OLDNIV  : INTEGER;  (* MEMORISE LE NIVEAU *)
  256  2  0     RPTRADR : ADRESSE;  (* ADR DU PTR RETOUR *)
  257  2  0     SVTADR  : ADRESSE;  (* ADR DU NOEUD SUIVANT *)
  258  2  0     TYP     : TYPEL;
  259  2  0
  260  2  0
  261  2  0   BEGIN
  262  2  1+    WRITELN ('       DEBUT ETABLI-CHEMIN');
  263  2  1     OLDNIV := NIVEAU;
  264  2  1     FIN := FALSE;
  265  2  1     CODENODE (CHEMIN^.DESCADR,CODE);   (*163*)
  266  2  1     ADRREPTR (CHEMIN^.DESCADR,RPTRADR,CODE);   (*177*)
  267  2  1     REPEAT
  268  2  2+      IF RPTRADR = NUL THEN FIN := TRUE   (*PLUS DE CHEMIN PARTANT DE LA*)
  269  2  2       ELSE BEGIN
  270  2  3+        ANALRPTR (RPTRADR,SVTADR,TYP,DESCADR);   (*193*)
  271  2  3         NIVNOEUD (DESCADR,NIVEAU);   (*211*)
  272  2  3         WRITELN ('NIV',NIVEAU);
  273  2  3         IF NIVEAU = 1 THEN BEGIN          (* C'EST UNE RACINE *)
  274  2  4+          CURCH := CHEMIN;
  275  2  4           CREERETAPE (CURCH,DESCADR,TYP);   (*104*)
  276  2  4           CREERRACINE (CURCH);   (*76*)
  277  2  4           END
  278  2  3-        ELSE IF (NIVEAU < OLDNIV) OR
  279  2  3                ((NIVEAU = OLDNIV) AND (NOT MEMORISE (CHEMIN,DESCADR)))   (*135*)
  280  2  3                THEN BEGIN
  281  2  4+          CURCH := CHEMIN;
  282  2  4           CREERETAPE (CURCH,DESCADR,TYP);   (*104*)
  283  2  4           ETABLICHEMIN (CURCH,NIVEAU);   (*261*)
  284  2  4           END;
  285  2  3-        RPTRADR := SVTADR;
  286  2  3         END;
  287  2  2-    UNTIL FIN;
  288  2  1-    WRITELN ('            FIN ETABLI CHEMIN');
  289  2  1   END;

END ETABLICH L=   834

  290  1  0-
  291  1  0   (*=================================================================*)
```

```
  292  1  0
  293  1  0   PROCEDURE CHOIXCHEMIN (RACINE:RACPTR; VAR RACHEMIN:RACPTR);
  294  2  0
  295  2  0   (*
  296  2  0 C   ENTREE    : RACINE   : PTR VERS LA PREMIERE RACINE
  297  2  0 C
  298  2  0 C   FONCTION  : PRESENTE A L'UTILISATEUR TOUS LES CHEMINS POSSIBLES POUR
  299  2  0 C                ATTEINDRE LA TABLE DESIREE ET LUI DEMANDE D'EN CHOISIR
  300  2  0 C                UN
  301  2  0 C
  302  2  0 C   SORTIE    : RACHEMIN : PTR VERS LE CHEMIN CHOISI PAR L'UTILISATEUR.
  303  2  0 C *)
  304  2  0
  305  2  0   VAR
  306  2  0     CORRECT : BOOLEAN;
  307  2  0     CPT     : INTEGER;
  308  2  0     CURAC   : RACPTR;     (* PTR VERS RACINE COURANTE *)
  309  2  0     ETPCOUR : ETAPTR;     (* PTR VERS ETAPE COURANTE *)
  310  2  0     NOM     : CH8;        (* NOM DU NOEUD *)
  311  2  0     NUMCHEM : INTEGER;    (* NUMERO DU CHEMIN *)
  312  2  0     REPONSE : INTEGER;
  313  2  0
  314  2  0
  315  2  0   BEGIN
  316  2  1+    IF RACINE^.PSUIV = NIL THEN WRITELN (' PAS DE CHEMIN ')
  317  2  1     ELSE BEGIN
  318  2  2+      NUMCHEM := 1;
  319  2  2       CURAC := RACINE^.PSUIV;
  320  2  2       WHILE CURAC <> NIL DO BEGIN       (* POUR TOUS LES CHEMINS *)
  321  2  3+        WRITE ('CHEMIN ',NUMCHEM,' : ');
  322  2  3         ETPCOUR := CURAC^.PETAPE;       (* PTR ETAPE COURANTE *)
  323  2  3         WHILE ETPCOUR <> NIL DO BEGIN   (* POUR TTES LES ETAPES DU CHEM *)
  324  2  4+          NAMENODE (ETPCOUR^.DESCADR,NOM);(* CHERCHE NOM DU NOEUD *)   (*225*)
  325  2  4           WRITE (' ',NOM);
  326  2  4           ETPCOUR := ETPCOUR^.PSUIV;      (* ETAPE SUIVANTE *)
  327  2  4         END;        (* ON EST A LA FIN D'UN CHEMIN *)
  328  2  3-        WRITELN;
  329  2  3         WRITELN;
  330  2  3         NUMCHEM := NUMCHEM + 1;
  331  2  3         CURAC := CURAC^.PSUIV;        (* RACINE SUIVANTE *)
  332  2  3       END;    (* IL N'Y A PLUS DE CHEMIN *)
  333  2  2-      WRITELN;
  334  2  2       CORRECT := FALSE;
  335  2  2       WHILE NOT CORRECT DO BEGIN
  336  2  3+        WRITELN ('QUEL NUMERO DE CHEMIN VOULEZ-VOUS ? : ');
  337  2  3         READ (REPONSE);
  338  2  3         IF REPONSE > NUMCHEM-1 THEN WRITELN ('IMPOSSIBLE')
  339  2  3         ELSE BEGIN
  340  2  4+          CORRECT := TRUE;
  341  2  4           RACHEMIN := RACINE;
  342  2  4           FOR CPT:= 1 TO REPONSE DO RACHEMIN := RACHEMIN^.PSUIV;
  343  2  4         END;
  344  2  3-      END;
  345  2  2-    END;
  346  2  1-  END;

END CHOIXCHE L=   630

  347  1  0-
  348  1  0   (*================================================================*)
```

```
 349  1  0
 350  1  0   PROCEDURE CHEMIN (VAR RACHEMIN: RACPTR);
 351  2  0
 352  2  0
 353  2  0   (*
 354  2  0 C   FONCTION : ETABLI TOUS LES CHEMINS POSSIBLES DU GRAPHE
 355  2  0 C              PERMETTANT D'ACCEDER A LA TABLE VISEE A PARTIR DE
 356  2  0 C              LA RACINE DU GRAPHE
 357  2  0 C              NB: LES CHEMINS SONT ETABLIS EN PARTANT DE LA TABLE
 358  2  0 C                  ET EN REMONTANT JUSQU'A LA RACINE
 359  2  0 C
 360  2  0 C   POSTCOND : RACHEMIN EST LE POINTEUR VERS LE CHEMIN CHOISI OU SA VALEUR
 361  2  0 C              EST NIL S'IL N'Y A PAS DE CHEMIN
 362  2  0 C
 363  2  0 C *)
 364  2  0
 365  2  0
 366  2  0
 367  2  0
 368  2  0   VAR
 369  2  0
 370  2  0     ADRNOEUD : ADRESSE;  (* ADRESSE DU NOEUD COURANT *)
 371  2  0     CHEMCUR  : ETAPTR;   (* PTR VERS ETAPE COURANTE DU CHEMIN *)
 372  2  0     CHEMIN   : ETAPTR;   (* PTR VERS LE CHEMIN *)
 373  2  0     I        : INTEGER;  (* COMPTEUR *)
 374  2  0     NIVEAU   : INTEGER;  (* NIVEAU ETAPE COURANTE *)
 375  2  0     NOMNOEUD : CH8;      (* NOM DU NOEUD QU'ON DESIRE ATTEINDRE *)
 376  2  0     RACINE   : RACPTR;   (* PTR VERS LA PREMIERE RACINE *)
 377  2  0     TYPELIEN : TYPEL;    (* TYPE DU LIEN ENTRE NOEUD *)
 378  2  0
 379  2  0
 380  2  0   BEGIN
 381  2  1+    WRITELN (' *******         DEBUT   ********* ');
 382  2  1     NEW (RACINE);                (* PREMIERE RACINE *)
 383  2  1     CURRAC := RACINE;
 384  2  1     (* ################  SAISIE DU NOEUD A ATTEINDRE ############ *)
 385  2  1     WRITELN ('QUEL NOEUD VOULEZ-VOUS ATTEINDRE ? (EN MAJUSCULES) :');
 386  2  1     READLN;
 387  2  1     FOR I := 1 TO 8 DO
 388  2  1       IF EOLN(INPUT) THEN NOMNOEUD[I] := ' '
 389  2  1       ELSE READ(NOMNOEUD[I]);
 390  2  1
 391  2  1     NODEDIC (NOMNOEUD,ADRNOEUD);   (*150*)
 392  2  1     IF ADRNOEUD = NUL THEN BEGIN
 393  2  2+     WRITELN ('CE NOEUD N''EXISTE PAS') ;
 394  2  2       RACHEMIN := NIL ;
 395  2  2       END
 396  2  1-    ELSE BEGIN
 397  2  2+      WRITELN ('CE NOEUD EXISTE');
 398  2  2       NIVNOEUD (ADRNOEUD,NIVEAU);   (*211*)
 399  2  2       WRITELN ('NIVEAU DU NOEUD :',NIVEAU);
 400  2  2       TYPELIEN := 0;
 401  2  2       CHEMCUR := NIL;
 402  2  2       CREERETAPE (CHEMCUR,ADRNOEUD,TYPELIEN);   (*104*)
 403  2  2       ETABLICHEMIN (CHEMCUR,NIVEAU);   (*261*)
 404  2  2       CHOIXCHEMIN (RACINE,RACHEMIN);   (*315*)
 405  2  2       END;
 406  2  1-    WRITELN (' *************    F I N  ************** ');
 407  2  1   END;

END CHEMIN   L=   814
```

```
411  1  0
412  1  0    PROCEDURE DUMPRAC (VAR ADRDMP : ADRESSE); EXTERNAL;
413  1  0
414  1  0    (*
415  1  0 C    PRECOND  :
416  1  0 C
417  1  0 C    FONCTION : ACCEDE A LA RACINE DU GRAPHE DANS LE DUMP ET RENVOI DANS
418  1  0 C               < ADRDMP > SON ADRESSE
419  1  0 C
420  1  0 C    POSTCOND : < ADRDMP > EST L'ADRESSE DE LA RACINE DANS LE DUMP OU BIEN
421  1  0 C               < ADRDMP > VAUT NUL SI ON NE L'A PAS TROUVE
422  1  0 C
423  1  0 C *)
424  1  0
425  1  0
426  1  0    (*================================================================*)
427  1  0
428  1  0    PROCEDURE INFOREFS ( POINT : ADRESSE; VAR TYP : CH2;
429  2  0                         VAR OFFSET : ADRESSE); EXTERNAL;
430  1  0
431  1  0    (*
432  1  0 C    PRECOND  : < POINT > EST L'ADRESSE DU DESCRIPTEUR DU POINTEUR DANS LE
433  1  0 C               GRAPHE.
434  1  0 C
435  1  0 C    FONCTION : RENVOI DANS < TYP > LE TYPE DU POINTEUR ET DANS < OFFSET >
436  1  0 C               SON OFFSET.
437  1  0 C
438  1  0 C    POSTCOND : < TYP > EST LE TYPE ET < OFFSET > EST L'OFFSET DU POINTEUR
439  1  0 C               DONT L'ADRESSE DU DESCRIPTEUR EST < ADRPTR >.
440  1  0 C *)
441  1  0
442  1  0    (*================================================================*)
443  1  0
444  1  0    PROCEDURE GETADR ( VAR ADR : ADRESSE ); EXTERNAL;
445  1  0
446  1  0    (*
447  1  0 C    PRECOND  : < ADR > EST L'ADRESSE DONT IL FAUT DONNER LE CONTENU
448  1  0 C
449  1  0 C    FONCTION : RENVOI DANS < ADR > LE CONTENU DE L'ADRESSE < ADR > DANS
450  1  0 C               LE DUMP
451  1  0 C
452  1  0 C    POSTCOND : < ADR > EST LE CONTENU DE L'ADRESSE < ADR > DU DUMP
453  1  0 C *)
454  1  0
455  1  0    (*================================================================*)
```

```
456  1  0
457  1  0   PROCEDURE DUMPREFS (VAR ADRDMP : ADRESSE;VAR ADRPTR : ADRESSE;
458  2  0                        VAR ERROR : INTEGER );
459  2  0
460  2  0   (*
461  2  0 C   PRECOND  : < ADRDMP > EST L'ADRESSE VALIDE DE LA TABLE COURANTE DANS
462  2  0 C               LE DUMP ET < ADRPTR > EST L'ADRESSE VALIDE DU DESCRIPTEUR
463  2  0 C               DU POINTEUR DANS LE GRAPHE VERS LA TABLE SUIVANTE.
464  2  0 C
465  2  0 C   FONCTION : APPLIQUE A LA TABLE DONT L'ADR EST < ADRDMP > LE PTR DONT
466  2  0 C               LA DESCRIPTION EST A L'ADR < ADRPTR > POUR TROUVER LA
467  2  0 C               TABLE SUIVANTE DANS LE DUMP ET RENVOI SON ADRESSE DANS
468  2  0 C               < ADRDMP >. SI ON NE TROUVE PAS LA TABLE ALORS ON INDIQUE
469  2  0 C               UNE ERREUR.
470  2  0 C
471  2  0 C   POSTCOND : SI ON TROUVE LA TABLE, < ADRDMP > EST L'ADRESSE DE CETTE
472  2  0 C               NOUVELLE TABLE COURANTE DANS LE DUMP. SINON < ERROR > A LA
473  2  0 C               VALEUR 3 ET < ADRDMP > A LA MEME VALEUR QU'EN ENTREE.
474  2  0 C *)
475  2  0
476  2  0   VAR
477  2  0
478  2  0     INTERADR : ADRESSE;                (* ADRESSE INTERMEDIAIRE *)
479  2  0     OFFSET   : ADRESSE;                (* OFFSET DU POINTEUR *)
480  2  0     TYP      : CH2;                    (* TYPE DU POINTEUR   *)
481  2  0
482  2  0   BEGIN
483  2  1+    WRITELN (' ACCES PAR REF SPL ');
484  2  1     INFOREFS ( ADRPTR, TYP , OFFSET );   (*428*)
485  2  1     WRITELN (' TYPE ',TYP);
486  2  1     WRITELN (' OFFSET ',OFFSET);
487  2  1     INTERADR := ADRDMP + OFFSET ;
488  2  1     GETADR (INTERADR);   (*444*)
489  2  1     IF INTERADR = 0 THEN ERROR := 3            (* TABLE PAS TROUVEE *)
490  2  1     ELSE ADRDMP := INTERADR;                   (* MISE A JOUR DE ADRDMP *)
491  2  1   END;

END DUMPREFS L=   372
                (* DUMPREFS *)
492  1  0-
493  1  0   (*=================================================================*)
494  1  0
495  1  0   PROCEDURE INFOVECT ( POINT : ADRESSE; VAR TYP : CH2;
496  2  0                        VAR OFFSET : ADRESSE; VAR NBRE : INTEGER ); EXTERNAL;
497  1  0
498  1  0   (*
499  1  0 C   PRECOND  : < POINT > EST L'ADRESSE DU DESCRIPTEUR DU VECTEUR DANS LE
500  1  0 C               GRAPHE.
501  1  0 C
502  1  0 C   FONCTION : RENVOI DANS < TYP > LE TYPE DU VECTEUR, DANS < OFFSET >
503  1  0 C               SON OFFSET ET DANS < NBRE > LE NOMBRE DE COMPOSANTS.
504  1  0 C
505  1  0 C   POSTCOND : < TYP > EST LE TYPE, < OFFSET > EST L'OFFSET ,ET < NBRE>
506  1  0 C             EST LE NBRE DE COMPOSANT DU VECTEUR DONT L'ADRESSE DU
507  1  0 C             DESCRIPTEUR EST < POINT >
508  1  0 C *)
509  1  0
510  1  0   (*=================================================================*)
```

```
  511  1  0
  512  1  0    PROCEDURE DUMPVECT (VAR ADRDMP : ADRESSE; VAR ADRPTR : ADRESSE;
  513  2  0                        VAR ERROR : INTEGER );
  514  2  0    (*
  515  2  0 C    PRECOND  : < ADRDMP > EST L'ADRESSE VALIDE DE LA TABLE COURANTE DANS
  516  2  0 C               LE DUMP ET < ADRPTR > EST L'ADRESSE VALIDE DU DESCRIPTEUR
  517  2  0 C               DU POINTEUR DANS LE GRAPHE VERS LA TABLE SUIVANTE.
  518  2  0 C
  519  2  0 C    FONCTION : APPLIQUE A LA TABLE DONT L'ADR EST < ADRDMP > LE PTR DONT
  520  2  0 C               LA DESCRIPTION EST A L'ADR < ADRPTR > POUR TROUVER LA
  521  2  0 C               TABLE SUIVANTE DANS LE DUMP ET RENVOI SON ADRESSE DANS
  522  2  0 C               < ADRDMP >.
  523  2  0 C
  524  2  0 C    POSTCOND : < ADRDMP > EST L'ADRESSE DE LA NOUVELLE TABLE COURANTE
  525  2  0 C               DANS LE DUMP.
  526  2  0 C *)
  527  2  0
  528  2  0    VAR
  529  2  0      CORRECT  : BOOLEAN;
  530  2  0      INTERADR : ADRESSE;          (* ADRESSE DE TRAVAIL *)
  531  2  0      J        : INTEGER;          (* COMPTEUR *)
  532  2  0      NCOMP    : INTEGER;          (* NBRE DE COMPOSANT DU VECTEUR *)
  533  2  0      OFFSET   : ADRESSE;          (* OFFSET DU VECTEUR *)
  534  2  0      POSVECT  : INTEGER;          (* POSITION DU VECTEUR SOUHAITE *)
  535  2  0      TYP      : CH2;              (* TYPE DU VECTEUR *)
  536  2  0
  537  2  0    BEGIN
  538  2  1+     WRITELN (' ACCES PAR VECTEUR ');
  539  2  1      INFOVECT (ADRPTR,TYP,OFFSET,NCOMP);   (*495*)
  540  2  1      WRITELN (' TYPE ',TYP);
  541  2  1      WRITELN ('OFFSET',OFFSET);
  542  2  1      WRITELN ('NBRE COMP ',NCOMP);
  543  2  1      CORRECT := FALSE;
  544  2  1      WHILE NOT CORRECT DO BEGIN
  545  2  2+       WRITELN ('QUELLE COMPOSANTE DU VECTEUR VOULEZ-VOUS ?  :');
  546  2  2        READ (POSVECT);
  547  2  2        IF POSVECT <= NCOMP THEN CORRECT := TRUE
  548  2  2        ELSE WRITELN (' $$$ IMPOSSIBLE, NOMBRE TROP ELEVE $$$ ');
  549  2  2        END;  (* WHILE *)
  550  2  1-     INTERADR := ADRDMP + OFFSET;
  551  2  1      FOR J := 2 TO POSVECT DO INTERADR := INTERADR +4;
  552  2  1      GETADR (INTERADR);   (*444*)
  553  2  1      IF INTERADR = 0 THEN ERROR := 3
  554  2  1      ELSE ADRDMP := INTERADR;
  555  2  1    END;

END DUMPVECT L=   658
                  (* DUMPVECT *)
  556  1  0-
  557  1  0    (*=================================================================*)
```

```
  558  1  0
  559  1  0    PROCEDURE INFOQUEU ( POINT : ADRESSE; VAR TYP : CH2;
  560  2  0                         VAR OFFSET : ADRESSE); EXTERNAL;
  561  1  0
  562  1  0    (*
  563  1  0 C     PRECOND   : < POINT > EST L'ADRESSE DU DESCRIPTEUR DE LA QUEUE DANS LE
  564  1  0 C                 GRAPHE.
  565  1  0 C
  566  1  0 C     FONCTION  : RENVOI DANS < TYP > LE TYPE DU POINTEUR ET DANS < OFFSET >
  567  1  0 C                 SON OFFSET.
  568  1  0 C
  569  1  0 C     POSTCOND  : < TYP > EST LE TYPE ET < OFFSET > EST L'OFFSET DU POINTEUR
  570  1  0 C                 DONT L'ADRESSE DU DESCRIPTEUR EST < ADRPTR >.
  571  1  0 C *)
  572  1  0
  573  1  0    (*=================================================================*)
  574  1  0
  575  1  0    PROCEDURE CHOIXTBL (VAR CODE : INTEGER; VAR POS : INTEGER );
  576  2  0
  577  2  0    (*
  578  2  0 C     PRECOND   :
  579  2  0 C
  580  2  0 C     FONCTION  : DEMANDE A L'UTILISATEUR LE CRITERE DE SELECTION DE LA TABLE
  581  2  0 C                 DANS UNE QUEUE OU UN ANNEAU ET RENVOI DANS < CODE > LE CODE
  582  2  0 C                 DU MODE DE SELECTION ET DANS < POS > LA POSITION DE LA TBL
  583  2  0 C
  584  2  0 C     POSTCOND  : < CODE > EST UN CODE DE SELECTION VALIDE.
  585  2  0 C *)
  586  2  0
  587  2  0    VAR
  588  2  0      CORRECT   : BOOLEAN;
  589  2  0
  590  2  0    BEGIN
  591  2  1+     WRITELN (' %%%%% CRITERE DE CHOIX :   %%%%%');
  592  2  1      WRITELN ('   1 - POSITION (IEME TABLE ) ');
  593  2  1      WRITELN ('   2 - DEPLACEMENT / VALEUR ');
  594  2  1      CORRECT := FALSE;
  595  2  1      WHILE NOT CORRECT DO BEGIN
  596  2  2+       WRITELN;
  597  2  2        WRITELN ('VOTRE CHOIX ? ');
  598  2  2        READ (CODE);
  599  2  2        IF CODE <= 1 THEN CORRECT := TRUE
  600  2  2        ELSE IF CODE = 2 THEN WRITELN ('PAS ENCORE DISPONIBLE ')
  601  2  2        ELSE WRITELN ('ERREUR CE CHOIX N''EXISTE PAS ');
  602  2  2        END; (* WHILE *)
  603  2  1-     CASE CODE OF
  604  2  2+       1 : BEGIN
  605  2  3+             WRITELN (' POSITION DE LA TABLE ? :');
  606  2  3              READ (POS);
  607  2  3            END;
  608  2  2-     END;  (* CASE *)
  609  2  1-   END;

END CHOIXTBL L=   518
                (* CHOIXTBL *)
  610  1  0-
  611  1  0    (*=================================================================*)
```

```
  612  1  0
  613  1  0   PROCEDURE SELECTBL ( VAR ADRDMP : ADRESSE; TYP : CH2; OFFSET : ADRESSE;
  614  2  0                            ADRFIN : ADRESSE; CODECH : INTEGER; POS : INTEGER;
  615  2  0                            VAR ERROR : INTEGER );
  616  2  0
  617  2  0   (*
  618  2  0 C   PRECOND  : < ADRDMP > EST L'ADRESSE DE LA TABLE COURANTE DANS LE DUMP,
  619  2  0 C              < TYP > EST LE TYPE DU POINTEUR VERS LA TABLE SUIVANTE,
  620  2  0 C              < OFFSET > EST L7OFFSET DE CE POINTEUR. < ADRFIN > EST LA
  621  2  0 C              VALEUR DE L'ADRESSE EN FIN DE QUEUE OU D'ANNEAU. < CODECH >
  622  2  0 C              EST LE CODE DE CHOIX A APPLIQUER, < POS > EST LA POSITION
  623  2  0 C              DE LA TABLE A CHERCHER DS LA QUEUE OU L'ANNEAU.
  624  2  0 C
  625  2  0 C   FONCTION : RECHERCHE DANS LA QUEUE OU L'ANNEAU DONT L7ADRESSE DE DEBUT
  626  2  0 C              EST < ADRDMP > LA TABLE DESIREE A L7AIDE DU CRITERE DE
  627  2  0 C              CHOIX < CODECH > ET DE LA POSITION < POS >. RENVOI DANS
  628  2  0 C              < ADRDMP > L'ADRESSE DE CETTE NOUVELLE TABLE SI ON L'A
  629  2  0 C              TROUVEE SINON < ADRDMP > GARDE SON ANCIENNE VALEUR ET
  630  2  0 C              < ERROR > VAUT 3
  631  2  0 C
  632  2  0 C   POSTCOND : SI ON A TROUVE LA TABLE DESIREE ALORS <ADRDMP> EST SON
  633  2  0 C              ADRESSE, SINON < ADRDMP > GARDE SA VALEUR D'ENTREE ET
  634  2  0 C              < ERROR > A LA VALEUR 3.
  635  2  0 C *)
  636  2  0
  637  2  0   VAR
  638  2  0     FIN      : BOOLEAN;
  639  2  0     I        : INTEGER;
  640  2  0     INTERADR : ADRESSE;              (* ADRESSE DE TRAVAIL *)
  641  2  0     TROUVE   : BOOLEAN;
  642  2  0
  643  2  0   BEGIN
  644  2  1+    WRITELN ('SELECTBL ');
  645  2  1     FIN := FALSE;
  646  2  1     INTERADR := ADRDMP;
  647  2  1     TROUVE := FALSE;
  648  2  1     I := 1;
  649  2  1     WHILE NOT FIN DO BEGIN
  650  2  2+      CASE CODECH OF
  651  2  3+        1 : IF I = POS THEN TROUVE := TRUE
  652  2  3              ELSE I := I + 1;
  653  2  3      END;  (* CASE *)
  654  2  2-      IF TROUVE THEN FIN := TRUE
  655  2  2      ELSE BEGIN
  656  2  3+        INTERADR := INTERADR + OFFSET;
  657  2  3         GETADR (INTERADR);                    (* ADR TABLE SUIVANTE *)   (*444*)
  658  2  3         IF INTERADR = ADRFIN THEN FIN := TRUE;
  659  2  3         END;  (* ELSE *)
  660  2  2-      END;  (* WHILE *)
  661  2  1-    IF NOT TROUVE THEN ERROR := 3
  662  2  1     ELSE ADRDMP := INTERADR;
  663  2  1     WRITELN (' SELECTBL FIN ');
  664  2  1   END;

END SELECTBL L=   390
                (* SELECTBL *)
  665  1  0-
  666  1  0   (*=================================================================*)
```

```
  667  1  0
  668  1  0   PROCEDURE DUMPQUEU (VAR ADRDMP : ADRESSE; VAR ADRPTR : ADRESSE;
  669  2  0                         VAR ERROR : INTEGER);
  670  2  0
  671  2  0   (*
  672  2  0 C   PRECOND  : < ADRDMP > EST L'ADRESSE VALIDE DE LA TABLE COURANTE DANS
  673  2  0 C              LE DUMP ET < ADRPTR > EST L'ADRESSE VALIDE DU DESCRIPTEUR
  674  2  0 C              DE LA QUEUE DANS LE GRAPHE VERS LA TABLE SUIVANTE.
  675  2  0 C
  676  2  0 C   FONCTION : APPLIQUE A LA TABLE DONT L'ADR EST < ADRDMP > LE PTR DONT
  677  2  0 C              LA DESCRIPTION EST A L'ADR < ADRPTR > POUR TROUVER LA
  678  2  0 C              TABLE SUIVANTE DANS LE DUMP ET RENVOI SON ADRESSE DANS
  679  2  0 C              < ADRDMP >.
  680  2  0 C
  681  2  0 C   POSTCOND : < ADRDMP > EST L'ADRESSE DE LA NOUVELLE TABLE COURANTE
  682  2  0 C              DANS LE DUMP.
  683  2  0 C *)
  684  2  0
  685  2  0   VAR
  686  2  0     ADRFIN   : ADRESSE;          (* VALEUR DE LA FIN DE QUEUE = 0 *)
  687  2  0     CODECH   : INTEGER;          (* CODE DU TYPE DE CHOIX DE LA TBL *)
  688  2  0     INTERADR : ADRESSE;          (* ADRESSE DE TRAVAIL *)
  689  2  0     OFFSET   : ADRESSE;          (* OFFSET DE LA QUEUE *)
  690  2  0     POS      : INTEGER;          (* POSITION DE LA TBL CHOISIE *)
  691  2  0     TYP      : CH2;              (* TYPE DU PTR DE LA QUEUE *)
  692  2  0
  693  2  0   BEGIN
  694  2  1+    WRITELN (' ACCES PAR QUEUE   ');
  695  2  1     INFOQUEU (ADRPTR,TYP,OFFSET);   (*559*)
  696  2  1     WRITELN ('TYPE ',TYP,' OFFSET ',OFFSET);
  697  2  1     CHOIXTBL (CODECH,POS);   (*590*)
  698  2  1     ADRFIN := 0;
  699  2  1     INTERADR := ADRDMP;
  700  2  1     SELECTBL (INTERADR,TYP,OFFSET,ADRFIN,CODECH,POS,ERROR);   (*643*)
  701  2  1     IF ERROR = 0 THEN ADRDMP := INTERADR;          (* PAS D'ERREUR *)
  702  2  1   END;

END DUMPQUEU L=   448
                (* DUMPQUEU *)
  703  1  0-
  704  1  0   (*=================================================================*)
  705  1  0
  706  1  0   PROCEDURE INFOANN ( POINT : ADRESSE; VAR TYP : CH2;
  707  2  0                         VAR OFFSET : ADRESSE); EXTERNAL;
  708  1  0
  709  1  0   (*
  710  1  0 C   PRECOND  : < POINT > EST L'ADRESSE DU DESCRIPTEUR DE L'ANNEAU DANS LE
  711  1  0 C              GRAPHE.
  712  1  0 C
  713  1  0 C   FONCTION : RENVOI DANS < TYP > LE TYPE DU POINTEUR ET DANS < OFFSET >
  714  1  0 C              SON OFFSET.
  715  1  0 C
  716  1  0 C   POSTCOND : < TYP > EST LE TYPE ET < OFFSET > EST L'OFFSET DU POINTEUR
  717  1  0 C              DONT L'ADRESSE DU DESCRIPTEUR EST < ADRPTR >.
  718  1  0 C *)
  719  1  0
  720  1  0   (*=================================================================*)
```

```
 721  1  0
 722  1  0   PROCEDURE DUMPRING (VAR ADRDMP : ADRESSE; VAR ADRPTR : ADRESSE;
 723  2  0                        VAR ERROR : INTEGER );
 724  2  0
 725  2  0   (*
 726  2  0 C   PRECOND  : < ADRDMP > EST L'ADRESSE VALIDE DE LA TABLE COURANTE DANS
 727  2  0 C              LE DUMP ET < ADRPTR > EST L'ADRESSE VALIDE DU DESCRIPTEUR
 728  2  0 C              DE L'ANNEAU DANS LE GRAPHE VERS LA TABLE SUIVANTE.
 729  2  0 C
 730  2  0 C   FONCTION : APPLIQUE A LA TABLE DONT L'ADR EST < ADRDMP > LE PTR DONT
 731  2  0 C              LA DESCRIPTION EST A L'ADR < ADRPTR > POUR TROUVER LA
 732  2  0 C              TABLE SUIVANTE DANS LE DUMP ET RENVOI SON ADRESSE DANS
 733  2  0 C              < ADRDMP >.
 734  2  0 C
 735  2  0 C   POSTCOND : < ADRDMP > EST L'ADRESSE DE LA NOUVELLE TABLE COURANTE
 736  2  0 C              DANS LE DUMP.
 737  2  0 C *)
 738  2  0
 739  2  0   VAR
 740  2  0     ADRFIN    : ADRESSE;              (* ADR FIN D'ANNEAU = ADR 1ERE TBL *)
 741  2  0     CODECH    : INTEGER;              (* CODE DU TYPE DE CHOIX DE LA TBL *)
 742  2  0     INTERADR  : ADRESSE;              (* ADRESSE DE TRAVAIL *)
 743  2  0     OFFSET    : ADRESSE;              (* OFFSET DE L'ANNEAU *)
 744  2  0     POS       : INTEGER;              (* POSITION DE LA TBL CHOISIE *)
 745  2  0     TYP       : CH2;                  (* TYPE DU PTR DE L'ANNEAU *)
 746  2  0
 747  2  0   BEGIN
 748  2  1+    WRITELN (' ACCES PAR ANNEAU  ');
 749  2  1     INFOANN (ADRPTR,TYP,OFFSET);   (*706*)
 750  2  1     WRITELN ('TYPE ',TYP,' OFFSET ',OFFSET);
 751  2  1     CHOIXTBL (CODECH,POS);   (*590*)
 752  2  1     ADRFIN := ADRDMP;
 753  2  1     INTERADR := ADRDMP;
 754  2  1     SELECTBL (INTERADR,TYP,OFFSET,ADRFIN,CODECH,POS,ERROR);   (*643*)
 755  2  1     IF ERROR = 0 THEN ADRDMP := INTERADR;          (* PAS D'ERREUR *)
 756  2  1
 757  2  1   END;

END DUMPRING L=   452
                (* DUMPRING *)
 758  1  0-
 759  1  0   (*=================================================================*)
 760  1  0
 761  1  0   PROCEDURE REFSPL (NODADR: ADRESSE; VAR PTR : ADRESSE; DEST : ADRESSE );
 762  2  0                      EXTERNAL;
 763  1  0
 764  1  0   (*
 765  1  0 C   PRECOND  : < NODADR > EST L'ADRESSE VALIDE D'UN NOEUD ET < DEST> EST
 766  1  0 C              L'ADR DU NOEUD VERS LEQUEL ON DOIT TROUVER UN PTR.
 767  1  0 C
 768  1  0 C   FONCTION : RENVOI DANS < PTR > L'ADRESSE D'UNE REFERENCE SIMPLE DU
 769  1  0 C              NOEUD D'ADRESSE < NODADR > VERS LE NOEUD DONT L' ADR EST
 770  1  0 C              < DEST>. SI ON N'A PAS TROUVE, < PTR > A LA VALEUR 'NUL'
 771  1  0 C
 772  1  0 C   POSTCOND : S'IL EXISTE UNE REFERENCE SIMPLE ENTRE LE NOEUD DONT
 773  1  0 C              L'ADRESSE EST < NODADR > ET LE NOEUD DONT L' ADR EST
 774  1  0 C              < DEST> ALORS < PTR > ETS L'ADRESSE DE CETTE REFERENCE
 775  1  0 C              SINON < PTR > VAUT 'NUL'
 776  1  0 C *)
 777  1  0
 778  1  0   (*=================================================================*)
```

```
  779  1  0
  780  1  0   PROCEDURE REFVECT (NODADR: ADRESSE; VAR PTR : ADRESSE; DEST : ADRESSE);
  781  2  0                      EXTERNAL;
  782  1  0
  783  1  0   (*
  784  1  0 C   PRECOND  : < NODADR > EST L'ADRESSE VALIDE D'UN NOEUD ET < DEST> EST
  785  1  0 C              L' ADR DU NOEUD VERS LEQUEL ON DOIT TROUVER UN VECTEUR.
  786  1  0 C
  787  1  0 C   FONCTION : RENVOI DANS < PTR > L'ADRESSE D'UN VECTEUR DU
  788  1  0 C              NOEUD D'ADRESSE < NODADR > VERS LE NOEUD DONT L' ADR EST
  789  1  0 C              < DEST>. SI ON N'A PAS TROUVE, < PTR > A LA VALEUR 'NUL'
  790  1  0 C
  791  1  0 C   POSTCOND : S'IL EXISTE UN VECTEUR ENTRE LE NOEUD DONT
  792  1  0 C              L'ADRESSE EST < NODADR > ET LE NOEUD DONT L' ADR EST
  793  1  0 C              < DEST> ALORS < PTR > EST L'ADRESSE DE CE VECTEUR
  794  1  0 C              SINON < PTR > VAUT 'NUL'
  795  1  0 C *)
  796  1  0
  797  1  0
  798  1  0   (*=================================================================*)
  799  1  0
  800  1  0   PROCEDURE AVANCE (TYP : TYPEL; VAR ADRPTR : ADRESSE;
  801  2  0                      VAR ETPCUR : ETAPTR; VAR NODCUR : ADRESSE );
  802  2  0
  803  2  0   (*
  804  2  0 C   PRECOND  : < TYP > EST LE TYPE DU LIEN ENTRE LE NOEUD COURANT
  805  2  0 C              < NODCUR > ET LE SUIVANT, < ETPCUR > EST L'ADR DE L'ETAPE
  806  2  0 C              COURANTE..
  807  2  0 C
  808  2  0 C   FONCTION : CHERCHE A L'AIDE DE < TYP > LE POINTEUR DU NOEUD
  809  2  0 C              COURANT VERS LE NOEUD SUIVANT ET RENVOI SON ADRESSE DANS
  810  2  0 C              < ADRPTR >, AVANCE < ETACUR > ET < NODCUR > VERS,RESPECTIV-
  811  2  0 C              EMENT L'ETAPE ET LE NOEUD SUIVANT.
  812  2  0 C
  813  2  0 C   POSTCOND : < ADRPTR > EST L'ADRESSE DU POINTEUR VERS LE NOUVEAU NOEUD
  814  2  0 C              COURANT DONT L'ADRESSE EST < NODCUR >. L'ADRESSE DE L'ETAPE
  815  2  0 C              CORRESPONDANTE SE TROUVE DANS < ETPCUR >.
  816  2  0 C              DANS LE CAS D'UNE QUEUE OU D'UN ANNEAU L'ADRESSE DU PTR
  817  2  0 C              < ADRPTR > EST EN FAIT L'ADRESSE DU DESCRIPTEUR DE LA QUEUE
  818  2  0 C              OU DE L' ANNEAU.
  819  2  0 C
  820  2  0 C *)
  821  2  0
  822  2  0   VAR
  823  2  0      NOM      : CH8;                  (* NOM DE LA TABLE A ATTEINDRE *)
  824  2  0
  825  2  0   BEGIN
  826  2  1+    ETPCUR := ETPCUR^.PSUIV;
  827  2  1     NAMENODE (ETPCUR^.DESCADR,NOM);   (*225*)
  828  2  1     WRITELN (' ACCES AU NOEUD : ',NOM);
  829  2  1     CASE TYP OF
  830  2  2+      63,64,65,66 : ADRPTR := NODCUR ;        (* QUEUE OU ANNEAU *)
  831  2  2       60          : REFSPL (NODCUR,ADRPTR,ETPCUR^.DESCADR);   (*761*)
  832  2  2       61          : REFVECT (NODCUR,ADRPTR,ETPCUR^.DESCADR);   (*780*)
  833  2  2       END; (* CASE *)
  834  2  1-    NODCUR := ETPCUR^.DESCADR;
  835  2  1   END;

END AVANCE   L=   476
                  (* AVANCE *)
  836  1  0-
  837  1  0   (*=================================================================*)
```

```
  838  1  0
  839  1  0   PROCEDURE DESCENTE (RACHEM :RACPTR);
  840  2  0
  841  2  0   (*
  842  2  0 C   PRECOND  : < RACHEM > EST LE POINTEUR VERS LA RACINE DU CHEMIN CHOISI
  843  2  0 C              OU < RACHEM > VAUT NIL S'IL N'Y A PAS DE CHEMIN.
  844  2  0 C
  845  2  0 C   FONCTION : PARCOURS LE CHEMIN ET ACCEDE AUX TABLES DANS LE DUMP
  846  2  0 C
  847  2  0 C *)
  848  2  0
  849  2  0
  850  2  0   VAR
  851  2  0
  852  2  0     ADRDMP   : ADRESSE;          (* ADRESSE COURANTE DANS LE DUMP *)
  853  2  0     ADRPTR   : ADRESSE;          (* ADRESSE DS GRAPHE DU PTR VERS ETP COUR*)
  854  2  0     ERROR    : INTEGER;          (* CODE DE L'ERREUR S'IL Y EN A UNE  *)
  855  2  0     ETPCUR   : ETAPTR;           (* ADRESSE DE L'ETAPE COURANTE *)
  856  2  0     NODCUR   : ADRESSE;          (* ADRESSE DU NOEUD COURANT *)
  857  2  0     NOMNOEUD : CH8;              (* NOM DU NOEUD COURANT *)
  858  2  0     TYPELIEN : TYPEL;            (* TYPE DU LIEN VERS NOEUD COURANT *)
  859  2  0
  860  2  0   BEGIN
  861  2  1+      IF RACHEM = NIL THEN WRITELN ('IL N''Y A PAS DE CHEMIN A PARCOURIR')
  862  2  1       ELSE BEGIN
  863  2  2+        ERROR := 0;                                (* PAS D'ERREUR         *)
  864  2  2         ETPCUR := RACHEM^.PETAPE;
  865  2  2         NODCUR := ETPCUR^.DESCADR;
  866  2  2         NAMENODE(NODCUR,NOMNOEUD);   (*225*)
  867  2  2         WRITELN ('ACCES AU NOEUD : ',NOMNOEUD);
  868  2  2         DUMPRAC (ADRDMP);   (*412*)
  869  2  2         WRITELN ('DUMPRAC FIN : ADRDMP =',ADRDMP);
  870  2  2         IF ADRDMP = 0   THEN ERROR := 1;     (* PAS TROUVE LA RACINE *)
  871  2  2         WHILE ( ETPCUR^.PSUIV <> NIL ) AND  (* TANT PAS FIN CHEMIN  *)
  872  2  2               ( ERROR = 0 ) DO BEGIN         (* ET PAS D'ERREUR .... *)
  873  2  3+           TYPELIEN := ETPCUR^.TYPELIEN ;     (* MEMOR. TYPE DE LIEN  *)
  874  2  3            AVANCE (TYPELIEN,ADRPTR,ETPCUR,NODCUR);   (*825*)
  875  2  3            IF ADRPTR = NUL THEN ERROR := 2   (* PAS TROUVE LE POINTEUR *)
  876  2  3            ELSE CASE TYPELIEN OF
  877  2  4+             60     : DUMPREFS(ADRDMP,ADRPTR,ERROR);   (*482*)
  878  2  4              61     : DUMPVECT(ADRDMP,ADRPTR,ERROR);   (*537*)
  879  2  4              63,64  : DUMPQUEU(ADRDMP,ADRPTR,ERROR);   (*693*)
  880  2  4              65,66  : DUMPRING(ADRDMP,ADRPTR,ERROR);   (*747*)
  881  2  4              END; (* ELSE *)
  882  2  3-           WRITELN ('ADRESSE TABLE COURANTE : ',ADRDMP);
  883  2  3            END; (* WHILE *)
  884  2  2-         WRITELN ('ERREUR CODE :',ERROR);
  885  2  2          IF ERROR <> 0 THEN ERREUR ( ERROR );   (*47*)
  886  2  2          END; (* ELSE *)
  887  2  1-      WRITELN;
  888  2  1   END;

END DESCENTE L=   994

  889  1  0-
  890  1  0   (*=============================================================================*)
  891  1  0
  892  1  0   BEGIN
  893  1  1+    CHEMIN (RACINE);   (*380*)
  894  1  1     WRITELN('PRINCIPAL');
  895  1  1     DESCENTE (RACINE);   (*860*)
  896  1  1     WRITELN ('############### F   I   N  ############');
  897  1  1   END.

END PARCOURS L=   180
```

```
FLAG LOCTN OBJECT CODE    ADDR1  ADDR2  STMNT M  SOURCE STATEMENT

   000000                                    1    INTERFAC START
 | 000000                                    2             ENTRY NODEDIC
                                             3             EXTRN BEGDIC
                                             4             EXTRN ENDIC
                                             5    *
                                             6    *        PERMET D'OBTENIR L'ADRESSE D'UN NOEUD A PARTIR DE SON NOM
                                             7    *
   000000 90 EC D00C                         8    NODEDIC  STM   R14,R12,12(R13)
   000004 05 A0                              9             BALR  R10,0
   000006                                   10             USING *,R10
   000000                                   11             USING PARNODE,R1          DSECT DES PARAM
   000006 58 20 1000      000000            12             L     R2,NOM              R2=ADR NOM
   00000A 58 30 A046      00004C            13             L     R3,ABEGDIC          R3=DEBUT DICTIO
   00000E 58 40 A04A      000050            14             L     R4,AENDDIC          R4=FIN DU DICTIO
   000012 58 50 1004      000004            15             L     R5,ADR
   000016 D5 07 30002000                    16    LOOP     CLC   0(8,R3),0(R2)
   00001C 47 80 A034      00003A            17             BE    FOUND               ON A TROUVE
   000020 41 30 300C                        18             LA    R3,12(0,R3)         ELT SUIVANT
   000024 19 34                             19             CR    R3,R4               EST-CE LA FIN ?
   000026 47 80 A028      00002E            20             BE    NOFOUND             OUI,ALORS ON N'A PAS TROUVE
   00002A 47 F0 A010      000016            21             B     LOOP                NON,ON TESTE LE SUIVANT
   00002E 58 60 A45A      000460            22    NOFOUND  L     R6,MOINSUN
   000032 50 65 0000      000000            23             ST    R6,0(R5)
   000036 47 F0 A03E      000044            24             B     FIN
   00003A 41 30 3008                        25    FOUND    LA    R3,8(0,R3)
   00003E D2 03 50003000                    26             MVC   0(4,R5),0(R3)       MOVE L'ADRESSE DU NOEUD
   000044                                   27    FIN      DS    0H
   000044 98 EC D00C                        28             LM    R14,R12,12(R13)
   000048 07 FE                             29             BR    R14
                                            30             DROP  R1
   00004C 00000000                          31    ABEGDIC  DC    A(BEGDIC)
   000050 00000000                          32    AENDDIC  DC    A(ENDIC)
                                            33    *
                                            34    *=====================================================
                                            35    *
   000054                                   36             ENTRY NIVNOEUD
                                            37    *
                                            38    *        PERMET D'OBTENIR LE NIVEAU D'UN NOEUD
                                            39    *
   000054 90 EC D00C                        40    NIVNOEUD STM   R14,R12,12(R13)
   000058 05 A0                             41             BALR  R10,0
   00005A                                   42             USING *,R10
   000000                                   43             USING PARNIV,R1           DSECT DES PARAM
   00005A 58 20 1004      000004            44             L     R2,NIVEAU           R2=ADR DU PARAM NIVEAU
   00005E 58 30 1000      000000            45             L     R3,NOEUD
   000062 58 33 0000      000000            46             L     R3,0(R3)            R3=ADR DU NOEUD
   000000                                   47             USING DTABL,R3
   000066 D2 01 A03A3002  000094  000002    48             MVC   NIV2,TABLEV
   00006C D5 01 A03AA03C  000094  000096    49             CLC   NIV2,NEG            EST-CE UN NIVEAU =  -1 ?
   000072 47 70 A026      000080            50             BNE   POSIT               NON,ALORS ON VA EN POSIT
   000076 D2 01 A038A03C  000092  000096    51             MVC   NIV,NEG             OUI,ALORS NIV = FFFF
   00007C 47 F0 A02C      000086            52             B     MOV
                                            53             DROP  R1,R3
   000080 D2 01 A038A03E  000092  000098    54    POSIT    MVC   NIV,POS             NIV = 0000
   000086 D2 03 2000A038          000092    55    MOV      MVC   0(4,R2),NIV         MOVE LE NIVEAU
```

```
FLAG LOCTN OBJECT CODE    ADDR1  ADDR2  STMNT M  SOURCE STATEMENT

   00008C 98 EC D00C                        56            LM    R14,R12,12(R13)
   000090 07 FE                             57            BR    R14
                                            58   *
   000092                                   59   NIV      DS    H
   000094                                   60   NIV2     DS    H
   000096 FFFF                              61   NEG      DC    H'-1'
   000098 0000                              62   POS      DC    H'0'
                                            63   *
                                            64   *================================================
                                            65   *
   00009A                                   66            ENTRY ADRREPTR
                                            67   *
                                            68   *        PERMET D'OBTENIR L'ADRESSE DU PREMIER POINTEUR RETOUR
                                            69   *        A PARTIR DE L'ADRESSE DU NOEUD ET DE SON CODE
                                            70   *
   00009A 90 EC D00C                        71   ADRREPTR STM   R14,R12,12(R13)
   00009E 05 A0                             72            BALR  R10,0
   0000A0                                   73            USING *,R10
   000000                                   74            USING PARREPTR,R1
   0000A0 58 20 1004      000004            75            L     R2,RPTR             R2=ADR DU PARAM RPTR
   0000A4 58 30 1000      000000            76            L     R3,DESCR            R3=ADR DE L'ADR DE LA TABLE
   0000A8 58 33 0000      000000            77            L     R3,0(R3)            R3=ADR DE LA TABLE
   0000AC 58 40 1008      000008            78            L     R4,COD              R4=ADR DU PARAM COD
   0000B0 D5 01 4000A06C          00010C    79            CLC   0(2,R4),RTAB
   0000B6 47 20 A024      0000C4            80            BH    REPT1               CE N'EST PAS UNE TABLE
   000000                                   81            USING DTABL,R3
   0000BA D2 03 2000301C          00001C    82            MVC   0(4,R2),TABRPTR     MOVE L'ADR DU RPTR
   0000C0 47 F0 A066      000106            83            B     REPT9
   0000C4 D5 01 4000A06E          00010E    84   REPT1    CLC   0(2,R4),RQS
   0000CA 47 20 A038      0000D8            85            BH    REPT2               CE N'EST PAS UNE QUEUE SIMPLE
   000000                                   86            USING DSQ,R3
   0000CE D2 03 20003018          000018    87            MVC   0(4,R2),SQRPTR      MOVE L'ADR DU RPTR
   0000D4 47 F0 A066      000106            88            B     REPT9
   0000D8 D5 01 4000A070          000110    89   REPT2    CLC   0(2,R4),RQD
   0000DE 47 20 A04C      0000EC            90            BH    REPT3               CE N'EST PAS UNE QUEUE DOUBLE
   000000                                   91            USING DDQ,R3
   0000E2 D2 03 2000301C          00001C    92            MVC   0(4,R2),DQRPTR      MOVE L'ADR DU RPTR
   0000E8 47 F0 A066      000106            93            B     REPT9
   0000EC D5 01 4000A072          000112    94   REPT3    CLC   0(2,R4),RAS
   0000F2 47 20 A060      000100            95            BH    REPT4               CE N'EST PAS UN ANNEAU SIMPLE
   000000                                   96            USING DSR,R3
   0000F6 D2 03 20003014          000014    97            MVC   0(4,R2),SRRPTR      MOVE L'ADR DU RPTR
   0000FC 47 F0 A066      000106            98            B     REPT9
   000100                                   99   REPT4    DS    0H
   000000                                  100            USING DDR,R3              C'EST UN ANNEAU DOUBLE
   000100 D2 03 20003018          000018   101            MVC   0(4,R2),DRRPTR      MOVE L'ADR DU RPTR
   000106 98 EC D00C                       102   REPT9    LM    R14,R12,12(R13)
   00010A 07 FE                            103            BR    R14
                                           104            DROP  R1,R3
                                           105   *
   00010C 0008                             106   RTAB     DC    H'8'
   00010E 000B                             107   RQS      DC    H'11'
   000110 000D                             108   RQD      DC    H'13'
   000112 000F                             109   RAS      DC    H'15'
                                           110   *
```

```
FLAG LOCTN OBJECT CODE    ADDR1  ADDR2  STMNT M  SOURCE STATEMENT
                                          111    *=================================================
                                          112    *
000114                                    113            ENTRY ANALRPTR
                                          114    *
                                          115    *       ANALYSE LE POINTEUR RETOUR (CFR SPECIF. PASCAL)
                                          116    *
000114 90 EC D00C                         117    ANALRPTR STM   R14,R12,12(R13)
000118 05 A0                              118             BALR  R10,0
00011A                                    119             USING *,R10
000000                                    120             USING PARANAL,R1          DSECT DES PARAM.
00011A 58 20 1004      000004             121             L     R2,SVT              R2=ADR DU PARAM. SVT
00011E 58 30 1008      000008             122             L     R3,TYP              R3=ADR DU PARAM. TYP
000122 58 40 100C      00000C             123             L     R4,NODE             R4=ADR DU PARAM. NODE
000126 58 50 1000      000000             124             L     R5,PTR              R5=ADR DU PARAM. PTR
00012A 58 55 0000      000000             125             L     R5,0(R5)            R5=ADR DU RPTR
000000                                    126             USING DRPTR,R5
00012E D2 03 2000500C         00000C      127             MVC   0(4,R2),RPTRNEXT
000134 D2 01 A0345000 00014E  000000      128             MVC   CODE2,RPTRCODE
00013A D2 03 3000A032         00014C      129             MVC   0(4,R3),CODE
000140 D2 03 40005004         000004      130             MVC   0(4,R4),RPTRTO
000146 98 EC D00C                         131             LM    R14,R12,12(R13)
00014A 07 FE                              132             BR    R14
                                          133             DROP  R1,R5
                                          134    *
00014C 0000                               135    CODE     DC    H'0'
00014E                                    136    CODE2    DS    H
                                          137    *
                                          138    *=================================================
                                          139    *
000150                                    140            ENTRY NAMENODE
                                          141    *
                                          142    *       PERMET D'OBTENIR LE NOM D'UN NOEUD A PARTIR DE SON ADRESSE
                                          143    *
000150 90 EC D00C                         144    NAMENODE STM   R14,R12,12(R13)
000154 05 A0                              145             BALR  R10,0
000156                                    146             USING *,R10
000000                                    147             USING PARNAME,R1          DSECT DES PARAM
000156 58 20 1000      000000             148             L     R2,ADRESS           R2=ADR DU PARAM ADR
00015A 58 30 1004      000004             149             L     R3,NAME             R3=ADR DU PARAM NOM
00015E 58 22 0000      000000             150             L     R2,0(R2)            R2=ADR DU NOEUD
000000                                    151             USING DTABL,R2
000162 58 40 2004      000004             152             L     R4,TABDIC           ADR DE L'ELT DS DIC
000166 D2 07 30004000                     153             MVC   0(8,R3),0(R4)
00016C 98 EC D00C                         154             LM    R14,R12,12(R13)
000170 07 FE                              155             BR    R14
                                          156             DROP  R1,R2
                                          157    *
                                          158    *=================================================
                                          159    *
000172                                    160            ENTRY CODENODE
                                          161    *
                                          162    *       PERMET D'OBTENIR LE CODE D'UN NOEUD A PARTIR DE SON ADRESSE
                                          163    *
000172 90 EC D00C                         164    CODENODE STM   R14,R12,12(R13)
000176 05 A0                              165             BALR  R10,0
```

```
FLAG LOCTN OBJECT CODE   ADDR1  ADDR2  STMNT M  SOURCE STATEMENT

 000178                                   166              USING *,R10
 000000                                   167              USING PARCODE,R1          DSECT DES PARAM
 000178 58 20 1000      000000            168              L     R2,DESCADR          R2=ADR DU PARAM DESCADR
 00017C 58 30 1004      000004            169              L     R3,CODE1            R3=ADR DU PARAM CODE
 000180 58 22 0000      000000            170              L     R2,0(R2)            R2=ADR DU NOEUD
 000184 D2 01 A0202000 000198             171              MVC   CO2,0(R2)           MOVE LE CODE
 00018A D2 01 3000A020        000198      172              MVC   0(2,R3),CO2
 000190 98 EC D00C                        173              LM    R14,R12,12(R13)
 000194 07 FE                             174              BR    R14
                                          175              DROP  R1
                                          176    *
 000196 0000                              177    CO1       DC    H'0'
 000198                                   178    CO2       DS    H
                                          179    *
                                          180    *=================================================
                                          181    *
 00019A                                   182              ENTRY REFSPL
                                          183    *
                                          184    *         PERMET D'OBTENIR L'ADRESSE DE LA REF.SIMPL
                                          185    *           ( CFR PASCAL )
                                          186    *
 00019A 90 EC D00C                        187    REFSPL    STM   R14,R12,12(R13)
 00019E 05 A0                             188              BALR  R10,0
 0001A0                                   189              USING *,R10
 000000                                   190              USING PARREFS,R1          DSECT DES PARAM
 0001A0 58 20 1000      000000            191              L     R2,NODADR           R2 = ADR DU PARAM NODADR
 0001A4 58 30 1004      000004            192              L     R3,ADRPTR           R3 = ADR DU PARAM ADRPTR
 0001A8 58 40 1008      000008            193              L     R4,DESTADR          R4 = ADR DU PARAM DEST
 0001AC 58 22 0000      000000            194              L     R2,0(R2)            R2 = ADR DU NOEUD
 000000                                   195              USING DTABL,R2
 0001B0 58 50 2010      000010            196              L     R5,TABPTR           ADRESSE 1ERE REFERENCE
 000000                                   197              USING DPTR,R5
 0001B4 D5 03 500C4000 00000C             198    COMP1     CLC   PTRTO,0(R4)         COMPARE AVEC DEST
 0001BA 47 80 A02A      0001CA            199              BE    REFS2               ON A TROUVE DONC FIN
 0001BE 58 50 5010      000010            200              L     R5,PTRNEXT          REFERENCE SUIVANTE
 0001C2 59 50 A2C0      000460            201              C     R5,MOINSUN          SI ON N'EST PAS A LA FIN
 0001C6 47 70 A014      0001B4            202              BNE   COMP1              ...ON RECOMMENCE
 0001CA 50 53 0000      000000            203    REFS2     ST    R5,0(R3)            RENVOI DE L'ADRESSE DE LA REFERENCE
 0001CE 98 EC D00C                        204              LM    R14,R12,12(R13)
 0001D2 07 FE                             205              BR    R14
                                          206              DROP  R1,R2,R5
                                          207    *
                                          208    *=================================================
                                          209    *
 0001D4                                   210              ENTRY REFVECT
                                          211    *
                                          212    *         PERMET D'OBTENIR L'ADRESSE DU VECTEUR
                                          213    *           ( CFR PASCAL )
                                          214    *
 0001D4 90 EC D00C                        215    REFVECT   STM   R14,R12,12(R13)
 0001D8 05 A0                             216              BALR  R10,0
 0001DA                                   217              USING *,R10
 000000                                   218              USING PARVECT,R1          DSECT DES PARAM
 0001DA 58 20 1000      000000            219              L     R2,NODADR2          R2 = ADR DU PARAM NODADR
 0001DE 58 30 1004      000004            220              L     R3,ADRPTR2          R3 = ADR DU PARAM ADRPTR
```

```
FLAG LOCTN OBJECT CODE     ADDR1  ADDR2  STMNT M  SOURCE STATEMENT

    0001E2 58 40 1008      000008           221          L     R4,DESTADR2         R4 = ADR DU PARAM DESTADR2
    0001E6 58 22' 0000     000000           222          L     R2,0(R2)            R2 = ADR DU NOEUD
    000000                                  223          USING DTABL,R2
    0001EA 58 50 2014      000014           224          L     R5,TABVEC           ADRESSE 1ER VECTEUR
    000000                                  225          USING DVECT,R5
    0001EE D5 03 50104000  000010           226  COMP2   CLC   VECTO,0(R4)        COMPARE AVEC DEST
    0001F4 47 80 A02A      000204           227          BE    VECT2               ON A TROUVE DONC FIN
    0001F8 58 50 5014      000014           228          L     R5,VECTNEXT         VECTEUR SUIVANT
    0001FC 59 50 A286      000460           229          C     R5,MOINSUN          SI ON N'EST PAS A LA FIN
    000200 47 70 A014      0001EE           230          BNE   COMP2              ...ON RECOMMENCE
    000204 50 53 0000      000000           231  VECT2   ST    R5,0(R3)            RENVOI DE L'ADRESSE DU VECTEUR
    000208 98 EC D00C                       232          LM    R14,R12,12(R13)
    00020C 07 FE                            233          BR    R14
                                            234          DROP  R1,R2,R5
                                            235  *
                                            236  *=================================================
                                            237  *
    00020E                                  238          ENTRY INFOREFS
                                            239  *
                                            240  *       PERMET D'OBTENIR LE TYPE ET L'OFFSET DE LA REF SIMPLE
                                            241  *
    00020E 90 EC D00C                       242  INFOREFS STM   R14,R12,12(R13)
    000212 05 A0                            243          BALR  R10,0
    000214                                  244          USING *,R10
    000000                                  245          USING PARINREF,R1         DSECT DES PARAM
    000214 58 20 1000      000000           246          L     R2,REFADR           R2 = ADR DU PARAM REFADR
    000218 58 30 1004      000004           247          L     R3,REFTYP           R3 = ADR DU PARAM REFTYP
    00021C 58 40 1008      000008           248          L     R4,REFOFF           R4 = ADR DU PARAM REFOFF
    000220 58 22 0000      000000           249          L     R2,0(R2)            R2 = ADR DE LA REF SIMPLE
    000000                                  250          USING DPTR,R2             DSECT DES REF SIMPLE
    000224 D2 01 30002002          000002   251          MVC   0(2,R3),PTRTYPE     MOVE LE TYPE
    00022A D2 03 40002004          000004   252          MVC   0(4,R4),PTROFFS     MOVE L'OFFSET
    000230 98 EC D00C                       253          LM    R14,R12,12(R13)
    000234 07 FE                            254          BR    R14
                                            255          DROP  R1,R2
                                            256  *
                                            257  *=================================================
                                            258  *
    000236                                  259          ENTRY INFOVECT
                                            260  *
                                            261  *       PERMET D'OBTENIR LE TYPE, L'OFFSET ET LE NBRE DE COMPOSANTS
                                            262  *       DU VECTEUR
                                            263  *
    000236 90 EC D00C                       264  INFOVECT STM   R14,R12,12(R13)
    00023A 05 A0                            265          BALR  R10,0
    00023C                                  266          USING *,R10
    000000                                  267          USING PARINVEC,R1         DSECT DES PARAM
    00023C 58 20 1000      000000           268          L     R2,VECADR           R2 = ADR DU PARAM ADRESSE
    000240 58 30 1004      000004           269          L     R3,VECTYP           R3 = ADR DU PARAM TYPE
    000244 58 40 1008      000008           270          L     R4,VECOFF           R4 = ADR DU PARAM OFFSET
    000248 58 50 100C      00000C           271          L     R5,VECNBR           R5 = ADR DU PARAM NBRE DE COMPOSANTS
    00024C 58 22 0000      000000           272          L     R2,0(R2)            R2 = ADR DU VECTEUR
    000000                                  273          USING DVECT,R2            DSECT DU VECTEUR
    000250 D2 01 30002002          000002   274          MVC   0(2,R3),VECTYPE     MOVE LE TYPE
    000256 D2 03 40002004          000004   275          MVC   0(4,R4),VECTOFFS    MOVE L'OFFSET
```

```
FLAG LOCTN OBJECT CODE    ADDR1   ADDR2  STMNT M  SOURCE STATEMENT

    00025C D2 03 50002008         000008   276            MVC   0(4,R5),VECTRAN     MOVE LE NBRE DE COMPOSANTS
    000262 98 EC D00C              000277   277            LM    R14,R12,12(R13)
    000266 07 FE                   000278   278            BR    R14
                                   000279   279            DROP  R1,R2
                                   000280   280 *
                                   000281   281 *=================================================
                                   000282   282 *
    000268                         000283   283            ENTRY INFOQUEU
                                   000284   284 *
                                   000285   285 *       PERMET D'OBTENIR LE TYPE ET L'OFFSET DE LA QUEUE
                                   000286   286 *
    000268 90 EC D00C              000287   287 INFOQUEU STM   R14,R12,12(R13)
    00026C 05 A0                   000288   288            BALR  R10,0
    00026E                         000289   289            USING *,R10
    000000                         000290   290            USING PARINQUE,R1          DSECT DES PARAM
    00026E 58 20 1000      000000  000291   291            L     R2,QUEADR            R2 = ADR DU PARAM QUEADR
    000272 58 30 1004      000004  000292   292            L     R3,QUETYP            R3 = ADR DU PARAM QUETYP
    000276 58 40 1008      000008  000293   293            L     R4,QUEOFF            R4 = ADR DU PARAM QUEOFF
    00027A 58 22 0000      000000  000294   294            L     R2,0(R2)             R2 = ADR DU DESCR DE QUEUE
    00027E D5 01 A0462000  0002B4  000295   295            CLC   DQUCODE,0(R2)        SI C'EST QUEUE DOUBLE UNIQUE
    000284 47 80 A034      0002A2  000296   296            BE    QUEDOUB
    000288 D5 01 A0482000  0002B6  000297   297            CLC   DQMCODE,0(R2)        SI C'EST QUEUE DOUBLE MULTIPLE
    00028E 47 80 A034      0002A2  000298   298            BE    QUEDOUB
    000000                         000299   299            USING DSQ,R2               C'EST UNE QUEUE SIMPLE
    000292 D2 01 3000200E         00000E   300            MVC   0(2,R3),SQLNKTYP     MOVE LE TYPE
    000298 D2 03 40002010         000010   301            MVC   0(4,R4),SQLNKOFF     MOVE L'OFFSET
    00029E 47 F0 A040      0002AE  000302   302            B     QUEFIN
    0002A2                         000303   303 QUEDOUB  DS    0H                   C'EST UNE QUEUE DOUBLE
    000000                         000304   304            USING DDQ,R2
    0002A2 D2 01 3000200C         00000C   305            MVC   0(2,R3),DQFORTY      MOVE LE TYPE DU FORWARD LINK
    0002A8 D2 03 40002010         000010   306            MVC   0(4,R4),DQFOROFF     MOVE L7OFFSET DU FORWARD LINK
    0002AE 98 EC D00C              000307   307 QUEFIN   LM    R14,R12,12(R13)
    0002B2 07 FE                   000308   308            BR    R14
                                   000309   309            DROP  R1,R2
                                   000310   310 *
    0002B4 000C                    000311   311 DQUCODE  DC    AL2(12)               CODE QUEUE DOUBLE UNIQUE
    0002B6 000D                    000312   312 DQMCODE  DC    AL2(13)               CODE QUEUE DOUBLE MULTIPLE
                                   000313   313 *
                                   000314   314 *=================================================
                                   000315   315 *
    0002B8                         000316   316            ENTRY INFOANN
                                   000317   317 *
                                   000318   318 *       PERMET D'OBTENIR LE TYPE ET L'OFFSET DE L'ANNEAU
                                   000319   319 *
    0002B8 90 EC D00C              000320   320 INFOANN  STM   R14,R12,12(R13)
    0002BC 05 A0                   000321   321            BALR  R10,0
    0002BE                         000322   322            USING *,R10
    000000                         000323   323            USING PARINANN,R1          DSECT DES PARAM
    0002BE 58 20 1000      000000  000324   324            L     R2,ANNADR            R2 = ADR DU PARAM ANNADR
    0002C2 58 30 1004      000004  000325   325            L     R3,ANNTYP            R3 = ADR DU PARAM ANNTYP
    0002C6 58 40 1008      000008  000326   326            L     R4,ANNOFF            R4 = ADR DU PARAM ANNOFF
    0002CA 58 22 0000      000000  000327   327            L     R2,0(R2)             R2 = ADR DU DESCR DE L'ANNEAU
    0002CE D5 01 A0462000  000304  000328   328            CLC   DAUCODE,0(R2)        SI C'EST ANNEAU DOUBLE UNIQUE
    0002D4 47 80 A034      0002F2  000329   329            BE    ANNDOUB
    0002D8 D5 01 A0482000  000306  000330   330            CLC   DAMCODE,0(R2)        SI C'EST ANNEAU DOUBLE MULTIPLE
```

```
FLAG LOCTN OBJECT CODE    ADDR1  ADDR2  STMNT M  SOURCE STATEMENT

    0002DE 47 80 A034     0002F2          331            BE    ANNDOUB
    000000                                332            USING DSR,R2                C'EST UN ANNEAU SIMPLE
    0002E2 D2 01 3000200E         00000E  333            MVC   0(2,R3),SRLNKTYP    MOVE LE TYPE
    0002E8 D2 03 40002010         000010  334            MVC   0(4,R4),SRLNKOFF    MOVE L'OFFSET
    0002EE 47 F0 A040     0002FE          335            B     ANNFIN
    0002F2                                336 ANNDOUB    DS    0H                  C'EST UN ANNEAU DOUBLE
    000000                                337            USING DDR,R2
    0002F2 D2 01 3000200C         00000C  338            MVC   0(2,R3),DRFORTY     MOVE LE TYPE DU FORWARD LINK
    0002F8 D2 03 40002010         000010  339            MVC   0(4,R4),DRFOROFF    MOVE L'OFFSET DU FORWARD LINK
    0002FE 98 EC D00C                     340 ANNFIN     LM    R14,R12,12(R13)
    000302 07 FE                          341            BR    R14
                                          342            DROP  R1,R2
                                          343 *
    000304 0010                           344 DAUCODE    DC    AL2(16)              CODE ANNEAU DOUBLE UNIQUE
    000306 0011                           345 DAMCODE    DC    AL2(17)              CODE ANNEAU DOUBLE MULTIPLE
                                          346 *
                                          347 *================================================
                                          348 *
                                          349 *          ENTRY DUMPRAC
                                          350 *
                                          351 *          PERMET D'OBTENIR L'ADRESSE DE LA RACINE DU DUMP
                                          352 *
                                          353 *DUMPRAC  STM   R14,R12,12(R13)
                                          354 *          BALR  R10,0
                                          355 *          USING *,R10
                                          356 *          LR    R3,R1                 SAUVE R1
                                          357 *          TAM   OPEN,
                                          358 *                FILE=$PRE.SLED.TEST,
                                          359 *                LINK=1
                                          360 *          TAM   ACCESS,
                                          361 *                LINK=1,
                                          362 *                KEY=XVT,
                                          363 *                LEN=12,
                                          364 *                DADDR=OUTAREA
                                          365 *          USING PARDRAC,R3            DSECT DES PARAM
                                          366 *          L     R2,ADRDMP             R2=ADR DU PARAM ADRDMP
                                          367 *          MVC   0(4,R2),24(R1)        MOVE L'ADRESSE DE LA RACINE
                                          368 *          LR    R1,R3                 RESTORE R1
                                          369 *          LM    R14,R12,12(R13)
                                          370 *          BR    R14
                                          371 *          DROP  R3
                                          372 *
                                          373 *
                                          374 *================================================
                                          375 *
    000308                                376            ENTRY DUMPRAC
                                          377 *
                                          378 *          PERMET D'OBTENIR L'ADRESSE DE LA RACINE DU DUMP
                                          379 *
    000308 90 EC D00C                     380 DUMPRAC    STM   R14,R12,12(R13)
    00030C 05 A0                          381            BALR  R10,0
    00030E                                382            USING *,R10
    00030E 18 31                          383            LR    R3,R1                 SAUVE R1
                                          384            CMD '%DUMPFILE','D1=$PRE.SLED.TEST'
                                        1 385            IDLKG ALIGN=C,VER=002
```

```
FLAG LOCTN OBJECT CODE    ADDR1  ADDR2  STMNT M  SOURCE STATEMENT

                                          386  2              *,VERSION 002
  000310                                  387  1         CNOP  0,4
  000310 45 10 A02A     000338            388  1         BAL   1,*+40
  000314 04                               389  1         DC    AL1(4)
  000315 000000                           390  1         DC    X'000000'
  000318 001F                             391  1         DC    AL2(31)
  00031A 4040                             392  1         DC    C'  '
  00031C 6CC4E4D4D7C6C9D3                 393  1         DC    C'%DUMPFILE'
  000325 40                               394  1         DC    C' '
  000326 C4F17E5BD7D9C54B                 395  1         DC    C'D1=$PRE.SLED.TEST'
  000338 0A 58                            396  1         SVC   88
                                          397             CMD '%D','E=D1.%XVT %XL4                    ',OUTAREA
                                          398  1         IDLKG ALIGN=C,VER=002
                                          399  2              *,VERSION 002
  00033A 0700                             400  1         CNOP  0,4
  00033C 45 10 A060     00036E            401  1         BAL   1,*+50
  000340 00                               402  1         DC    AL1(0)
  000341 000464                           403  1         DC    AL3(OUTAREA)
  000344 0029                             404  1         DC    AL2(41)
  000346 4040                             405  1         DC    C'  '
  000348 6CC4                             406  1         DC    C'%D'
  00034A 40                               407  1         DC    C' '
  00034B C57EC4F14B6CE7E5                 408  1         DC    C'E=D1.%XVT %XL4                      '
  00036E 0A 58                            409  1         SVC   88
  000000      '                           410             USING PARDRAC,R3          DSECT DES PARAM
  000370 58 20 3000     000000            411             L     R2,ADRDMP           R2=ADR DU PARAM ADRDMP
  000374 D2 07 A092A1B3 0003A0 0004C1     412             MVC   DUMPR2,OUTAREA+X'5D'  MOVE L'ADRESSE DE LA RACINE
  00037A DC 07 A092A35A 0003A0 000668     413             TR    DUMPR2,TRTBL
  000380 F2 47 A08DA092 00039B 0003A0     414             PACK  DUMPR1,DUMPR2
  000386 F1 44 A08DA08D 00039B 00039B     415             MVO   DUMPR1,DUMPR1       MOVE L'ADRESSE DS LE PARAM
  00038C D2 03 2000A08D         00039B    416             MVC   0(4,R2),DUMPR1      MOVE L'ADRESSE DS LE PARAM
  000392 18 13                            417             LR    R1,R3               RESTORE R1
  000394 98 EC D00C                       418             LM    R14,R12,12(R13)
  000398 07 FE                            419             BR    R14
                                          420             DROP  R3
                                          421  *
  00039A                                  422             DS    C
  00039B                                  423  DUMPR1     DS    CL5
  0003A0                                  424  DUMPR2     DS    CL8
                                          425  *
                                          426  *================================================
                                          427  *
                                          428  *         ENTRY GETADR
                                          429  *
                                          430  *         PERMET D'OBTENIR LE CONTENU DE L'ADRESSE DS DUMP
                                          431  *
                                          432  *GETADR   STM   R14,R12,12(R13)
                                          433  *         BALR  R10,0
                                          434  *         USING *,R10
                                          435  *         LR    R3,R1               SAUVE R1
                                          436  *         USING PARGETA,R3          DSECT DES PARAM
                                          437  *         L     R2,GETAD            R2 = ADR DU PARAM ADRESSE
                                          438  *         LA    R4,TAM10005         R4 = ADR DC TAM
                                          439  *         LA    R4,24(R4)           R4 = ADR DE L'ADRESSE DANS TAM
                                          440  *         MVC   0(4,R4),0(R2)       MOVE L'ADRESSE
```

```
FLAG LOCTN OBJECT CODE    ADDR1  ADDR2  STMNT M  SOURCE STATEMENT
                                          441    *        TAM   ACCESS,
                                          442    *              ADDR=X'501000',
                                          443    *              DADDR=OUTAREA,
                                          444    *              LEN=4,
                                          445    *              LINK=1
                                          446    *        MVC   0(4,R2),OUTAREA     MOVE LE CONTENU
                                          447    *        LR    R1,R3               RESTORE R1
                                          448    *        LM    R14,R12,12(R13)
                                          449    *        BR    R14
                                          450    *        DROP  R3
                                          451    *
                                          452    *
                                          453    *================================================
                                          454    *
0003A8                                    455           ENTRY GETADR
                                          456    *
                                          457    *        PERMET D'OBTENIR LE CONTENU DE L'ADRESSE DS DUMP
                                          458    *
0003A8 90 EC D00C                         459    GETADR  STM   R14,R12,12(R13)
0003AC 05 A0                              460           BALR  R10,0
0003AE                                    461           USING *,R10
0003AE 18 31                              462           LR    R3,R1               SAUVE R1
000000                                    463           USING PARGETA,R3          DSECT DES PARAM
0003B0 58 20 3000      000000             464           L     R2,GETAD            R2 = ADR DU PARAM ADRESSE
0003B4 D2 03 A0962000  000444             465           MVC   GETA1,0(R2)         MOVE L'ADRESSE DS GETA1
0003BA 41 90 0008      000008             466           LA    R9,8                R9 = COMPTEUR
0003BE 58 60 A096      000444             467           L     R6,GETA1            R6 = SOURCE
0003C2 41 50 A0A1      00044F             468           LA    R5,GETA2+7          R5 = PTR COURANT DS DESTINAT.
0003C6 8C 60 0004      000004             469    GETALOOP SRDL  R6,4                MOVE 1/2 BYTE A DROITE
0003CA 88 70 001C      00001C             470           SRL   R7,28               R7 = DERNIER 1/2 BYTE COURANT
0003CE 41 87 A3B4      000762             471           LA    R8,TRTBL2(R7)       R8 = ADRESSE DS TBL
0003D2 D2 00 50008000                     472           MVC   0(1,R5),0(R8)       MOVE LE CARACT
0003D8 06 50                              473           BCTR  R5,R0               R5 = R5-1
0003DA 46 90 A018      0003C6             474           BCT   R9,GETALOOP         SI ON N'A PAS FINI ON RECOMMENCE
0003DE 41 40 A03A      0003E8             475           LA    R4,GETAIDDC         ADR DE LA DC DE AID
0003E2 D2 07 4016A09A         000448      476           MVC   22(8,R4),GETA2      MOVE L'ADRESSE DS LA DC
                                          477    GETAIDDC CMD   '%D','E=D1.V''00501000'' %XL4          ',OUTAREA
                                          478  1         IDLKG ALIGN=C,VER=002
                                          479  2               *,VERSION 002
0003E8                                    480  1         CNOP  0,4
0003E8 45 10 A068      000416             481  1 GETAIDDC BAL   1,*+46
0003EC 00                                 482  1         DC    AL1(0)
0003ED 000464                             483  1         DC    AL3(OUTAREA)
0003F0 0026                               484  1         DC    AL2(38)
0003F2 4040                               485  1         DC    C'  '
0003F4 6CC4                               486  1         DC    C'%D'
0003F6 40                                 487  1         DC    C' '
0003F7 C57EC4F14BE57DF0                   488  1         DC    C'E=D1.V''00501000'' %XL4          '
000416 0A 58                              489  1         SVC   88
000418 D2 07 A0A8A125 000456 0004D3       490           MVC   GETA5,OUTAREA+X'6F'  MOVE LE CONTENU
00041E DC 07 A0A8A2BA 000456 000668       491           TR    GETA5,TRTBL
000424 F2 47 A0A3A0A8 000451 000456       492           PACK  GETA4,GETA5
00042A F1 44 A0A3A0A3 000451 000451       493           MVO   GETA4,GETA4
000430 D2 00 A0A3A0A2 000451 000450       494           MVC   GETA4(1),GETA3      NEUTRALISE LE PREMIER BYTE
000436 D2 03 2000A0A3         000451      495           MVC   0(4,R2),GETA4       RENVOI LE CONTENU
```

```
FLAG LØCTN ØBJECT CØDE    ADDR1  ADDR2  STMNT M  SØURCE STATEMENT

    00043C 18 13                          496            LR    R1,R3                 RESTØRE R1
    00043E 98 EC D00C                     497            LM    R14,R12,12(R13)
    000442 07 FE                          498            BR    R14
                                          499            DRØP  R3
                                          500    *
    000444                                501    GETA1   DS    A
    000448                                502    GETA2   DS    CL8
    000450 00                             503    GETA3   DC    X'00'
    000451                                504    GETA4   DS    CL5
    000456                                505    GETA5   DS    CL8
                                          506    *
                                          507    *
                                          508    *******************************************************
                                          509    *
    000460 FFFFFFFF                       510    MØINSUN DC    F'-1'
                                          511    *ØUTAREA  DS    0F
                                          512    *         DC    512C' '
    000464                                513    ØUTAREA DS    0F
    000464 0190                           514            DC    AL2(400)
    000466 4040                           515            DC    C'  '
    000468 40                             516            DC    512C' '
                                          517    *
    000668 F0F0                           518    TRTBL   DC    256'00'
                          000729          519            ØRG   TRTBL+X'C1'
    000729 0A0B0C0D0E0F                   520            DC    X'0A0B0C0D0E0F'
                          000758          521            ØRG   TRTBL+X'F0'
    000758 0001020304050607               522            DC    X'00010203040506070809'
                                          523    *
    000762 F0F1F2F3F4F5F6F7               524    TRTBL2  DC    C'0123456789ABCDEF'
                                          525    *
                                          526    *******************************************************
                                          527    *       D S E C T       (PARAMETRES)                  *
                                          528    *******************************************************
    000000                                529    PARNØDE DSECT
    000000                                530    NØM     DS    A
    000004                                531    ADR     DS    A
                                          532    *
    000000                                533    PARNIV  DSECT
    000000                                534    NØEUD   DS    A
    000004                                535    NIVEAU  DS    A
                                          536    *
    000000                                537    PARREPTR DSECT
    000000                                538    DESCR   DS    A
    000004                                539    RPTR    DS    A
    000008                                540    CØD     DS    A
                                          541    *
    000000                                542    PARANAL DSECT
    000000                                543    PTR     DS    A
    000004                                544    SVT     DS    A
    000008                                545    TYP     DS    A
    00000C                                546    NØDE    DS    A
                                          547    *
    000000                                548    PARNAME DSECT
    000000                                549    ADRESS  DS    A
    000004                                550    NAME    DS    A
```

```
FLAG LOCTN OBJECT CODE    ADDR1  ADDR2  STMNT M  SOURCE STATEMENT
                                          551    *
000000                                    552    PARCODE  DSECT
000000                                    553    DESCADR  DS     A
000004                                    554    CODE1    DS     A
                                          555    *
000000                                    556    PARREFS  DSECT
000000                                    557    NODADR   DS     A
000004                                    558    ADRPTR   DS     A
000008                                    559    DESTADR  DS     A
                                          560    *
000000                                    561    PARVECT  DSECT
000000                                    562    NODADR2  DS     A
000004                                    563    ADRPTR2  DS     A
000008                                    564    DESTADR2 DS     A
                                          565    *
000000                                    566    PARDRAC  DSECT
000000                                    567    ADRDMP   DS     A
                                          568    *
000000                                    569    PARINREF DSECT
000000                                    570    REFADR   DS     A
000004                                    571    REFTYP   DS     A
000008                                    572    REFOFF   DS     A
                                          573    *
000000                                    574    PARINVEC DSECT
000000                                    575    VECADR   DS     A
000004                                    576    VECTYP   DS     A
000008                                    577    VECOFF   DS     A
00000C                                    578    VECNBR   DS     A
                                          579    *
000000                                    580    PARINQUE DSECT
000000                                    581    QUEADR   DS     A
000004                                    582    QUETYP   DS     A
000008                                    583    QUEOFF   DS     A
                                          584    *
000000                                    585    PARINANN DSECT
000000                                    586    ANNADR   DS     A
000004                                    587    ANNTYP   DS     A
000008                                    588    ANNOFF   DS     A
                                          589    *
000000                                    590    PARGETA  DSECT
000000                                    591    GETAD    DS     A
                                          592    *
                                          593    *#########################################################*
                                          594    *                    D S E C T                          *
                                          595    *#########################################################*
                                          596    *
                                          597    ************************
                                          598    *       TABLE          *
                                          599    ************************
                                          600    *
000000                                    601    DTABL    DSECT
000000                                    602    TABCODE  DS     AL2              TABLE CODE
000002                                    603    TABLEV   DS     AL2              TABLE LEVEL
000004                                    604    TABDIC   DS     A                A OF DIC ELT
000008                                    605    TABLEN   DS     A                TABLE LENGHT
```

```
FLAG LOCTN OBJECT CODE    ADDR1  ADDR2  STMNT M  SOURCE STATEMENT
    00000C                              606     TABLTYPE DS    AL2          TABLE LENGHT TYPE
                          00D3F1        607     LTYPE1   EQU   C'L1'        1 BYTE
                          00D3F2        608     LTYPE2   EQU   C'L2'        2 BYTES
                          00D3F3        609     LTYPE3   EQU   C'L3'        3 BYTES
                          00D3F4        610     LTYPE4   EQU   C'L4'        4 BYTES
                          000000        611     LTYPECT  EQU   X'00'        LENGHT IS CONSTANT
    00000E                              612              DS    AL2          UNUSED
    000010                              613     TABPTR   DS    A            A OF PTR QUEUE
    000014                              614     TABVEC   DS    A            A OF VEC
    000018                              615     TABMPTR  DS    A            A OF MPTR
    00001C                              616     TABRPTR  DS    A            A OF RPTR
                                        617     *
                                        618     *************************
                                        619     *    REFERENCE SIMPLE   *
                                        620     *************************
                                        621     *
    000000                              622     DPTR     DSECT
    000000                              623     PTRCODE  DS    AL2          PTR CODE
                          00001E        624     REVERSE  EQU   X'1E'        REVERSE EXISTE
                          00001F        625     NOREV    EQU   X'1F'        REVERSE N'EXISTE PAS
    000002                              626     PTRTYPE  DS    CL2          PTR TYPE
    000004                              627     PTROFFS  DS    A            OFFSET OF PTR IN TABLE
    000008                              628     PTRFROM  DS    A            ADR DESCR OF ORIGIN
    00000C                              629     PTRTO    DS    A            ADR DESCR OF EXTREMITY
    000010                              630     PTRNEXT  DS    A            ADR OF NEXT PTR IN TABLE
                                        631     *
                                        632     *************************
                                        633     *       VECTEUR         *
                                        634     *************************
                                        635     *
    000000                              636     DVECT    DSECT
    000000                              637     VECTCODE DS    AL2          VECTOR CODE
                          000028        638     VCODE    EQU   X'28'
    000002                              639     VECTYPE  DS    CL2          VECTOR TYPE
    000004                              640     VECTOFFS DS    A            VECTOR OFFSET (1ST COMP)
    000008                              641     VECTRAN  DS    A            VECTOR RANGE
    00000C                              642     VECTFROM DS    A            ADR OF DESCR ORIGIN
    000010                              643     VECTO    DS    A            ADR OF DESCR EXTREMITY
    000014                              644     VECTNEXT DS    A            NEXT VECTOR OF TABLE
                                        645     *
                                        646     *************************
                                        647     *    POINTEUR RETOUR    *
                                        648     *************************
                                        649     *
    000000                              650     DRPTR    DSECT
    000000                              651     RPTRCODE DS    AL2          RPTRCODE
                          00003C        652     RPTRPTR  EQU   X'3C'        REVERSE OF PTR
                          00003D        653     RPTRVECT EQU   X'3D'                  VECT
                          00003E        654     RPTRMPTR EQU   X'3E'                  MPTR
                          00003F        655     RPTRSQ   EQU   X'3F'                  SQ
                          000040        656     RPTRDQ   EQU   X'40'                  DQ
                          000041        657     RPTRSR   EQU   X'41'                  SR
                          000042        658     RPTRDR   EQU   X'42'                  DR
    000002                              659              DS    AL2          UNUSED
    000004                              660     RPTRTO   DS    A            A. OF DESC OF EXTREMITY
```

```
FLAG LOCTN OBJECT CODE    ADDR1  ADDR2  STMNT M  SOURCE STATEMENT
    000008                              661     RPTRFROM DS    A                A. OF DESC OF ORIGIN
    00000C                              662     RPTRNEXT DS    A                A. OF NEXT RPTR
                                        663     *
                                        664     ************************
                                        665     *     SIMPLE QUEUE     *
                                        666     ************************
                                        667     *
    000000                              668     DSQ      DSECT
    000000                              669     SQCODE   DS    AL2              CODE OF SQ
                        00000A          670     SQUNI    EQU   X'0A'
                        00000B          671     SQNOUNI  EQU   X'0B'
    000002                              672     SQLEV    DS    AL2              LEVEL OF SQ
    000004                              673     SQDIC    DS    A                ADR OF DIC ELT
    000008                              674     SQNODE   DS    A                ADR OF DESCR OF QUEUE ELT
    00000C                              675              DS    AL2              UNUSED
    00000E                              676     SQLNKTYP DS    CL2              LINKTYPE
    000010                              677     SQLNKOFF DS    A                OFFSET OF LINK
    000014                              678     SQEOQ    DS    A                END OF QUEUE VALUE
    000018                              679     SQRPTR   DS    A                ADR OF RPTR OF SQ
                                        680     *
                                        681     ************************
                                        682     *     SIMPLE ANNEAU    *
                                        683     ************************
                                        684     *
    000000                              685     DSR      DSECT
    000000                              686     SRCODE   DS    AL2              CODE OF SR
                        00000E          687     SRUNI    EQU   X'0E'
                        00000F          688     SRNOUNI  EQU   X'0F'
    000002                              689     SRLEV    DS    AL2
    000004                              690     SRDIC    DS    A                ADR OF DIC ELT
    000008                              691     SRNODE   DS    A                ADR OF DESCR OF ELT
    00000C                              692              DS    AL2              UNUSED
    00000E                              693     SRLNKTYP DS    CL2
    000010                              694     SRLNKOFF DS    A                OFFSET OF LINK
    000014                              695     SRRPTR   DS    A                ADR OF RPTR DESCRIPT
                                        696     *
                                        697     ************************
                                        698     *     DOUBLE QUEUE     *
                                        699     ************************
                                        700     *
    000000                              701     DDQ      DSECT
    000000                              702     DQCODE   DS    AL2              CODE OF DQ
                        00000C          703     DQUNI    EQU   X'0C'
                        00000D          704     DQNOUNI  EQU   X'0D'
    000002                              705     DQLEV    DS    AL2              LEVEL OF DQ
    000004                              706     DQDIC    DS    A                ADR OF DIC ELT
    000008                              707     DQNODE   DS    A                ADR OF ELT DESCR
    00000C                              708     DQFORTY  DS    CL2              TYPE OF FORWARD LINK
    00000E                              709     DQBACTY  DS    CL2              TYPE OF BACKWARD LINK
    000010                              710     DQFOROFF DS    A                OFFSET OF FORWARD LINK
    000014                              711     DQBACOFF DS    A                OFFSET OF BACKWARD LINK
    000018                              712     DQEOQ    DS    A                END OF QUEUE CODE
    00001C                              713     DQRPTR   DS    A                ADR OF RPTR DESCR
                                        714     *
                                        715     ************************
```

```
FLAG LØCTN ØBJECT CØDE    ADDR1  ADDR2  STMNT M  SØURCE STATEMENT

                                        716     *     DØUBLE ANNEAU     *
                                        717     *************************
                                        718     *
000000                                  719     DDR       DSECT
000000                                  720     DRCØDE    DS     AL2              CØDE ØF DR
                          000010        721     DRUNI     EQU    X'10'
                          000011        722     DRNØUNI   EQU    X'11'
000002                                  723     DRLEV     DS     AL2              LEVEL ØF DR
000004                                  724     DRDIC     DS     A                ADR ØF DIC ELT
000008                                  725     DRNØDE    DS     A                ADR ØF ELT DESCR
00000C                                  726     DRFØRTY   DS     CL2              TYPE ØF FØRWARD LINK
00000E                                  727     DRBACTY   DS     CL2              TYPE ØF BACKWARD LINK
000010                                  728     DRFØRØFF  DS     A                ØFFSET ØF FØRWARD LINK
000014                                  729     DRBACØFF  DS     A                ØFFSET ØF BACKWARD LINK
000018                                  730     DRRPTR    DS     A                ADR ØR RPTR
                                        731     *
                                        732     *
                          000000        733     R0        EQU    0
                          000001        734     R1        EQU    1
                          000002        735     R2        EQU    2
                          000003        736     R3        EQU    3
                          000004        737     R4        EQU    4
                          000005        738     R5        EQU    5
                          000006        739     R6        EQU    6
                          000007        740     R7        EQU    7
                          000008        741     R8        EQU    8
                          000009        742     R9        EQU    9
                          00000A        743     R10       EQU    10
                          00000B        744     R11       EQU    11
                          00000C        745     R12       EQU    12
                          00000D        746     R13       EQU    13
                          00000E        747     R14       EQU    14
                          00000F        748     R15       EQU    15
                          000010        749     R16       EQU    16
                                        750               END

FLAGS IN 00000 STATEMENTS, 000 PRIVILEGED FLAGS, 000 MNØTES

HIGHEST ERRØR-WEIGHT : -

THIS PRØGRAM WAS ASSEMBLED BY THE SIEMENS ASSEMBLER (F) V29.1X01   CØRR LEVEL: -
USER MACRØLIBRARY :    MAR.LIB
SYSTEM MACRØLIBRARY :  $BASE.LIB.M.SYSLIB.859J

ASSEMBLY TIME :        5.8824  SEC.
```