MASTER EN SCIENCES INFORMATIQUES

AIDE : Un logiciel de documentation en ligne interactif

Bokor, Karoly

Award date:
1984

Awarding institution:
Universite de Namur

Facultés universitaires Notre Dame de la Paix, Namur

Institut d'Informatique

AIDE :

UN LOGICIEL DE DOCUMENTATION

EN LIGNE INTERACTIF

Mémoire présenté par Károly BOKOR

en vue de l'obtention du grade de

Licencié et Maître en Informatique

Année académique 1983-1984

Remerciements

Je tiens à remercier Monsieur Michel DEBAR et le Père Jacques BERLEUR qui m'ont aidé et conseillé pour la réalisation de ce mémoire.

Je voudrais aussi remercier ceux qui m'ont accueilli pour la durée de mon stage : Monsieur Claude BARBE, de la firme SCHLUMBERGER, de Paris; Messieurs Pierre GRAVIER, Alain et Pascal CHESNAIX, du Centre Mondial pour l'Informatique et les Ressources Humaines, de Paris; et enfin, tout particulièrement, le personnel du Centre de Calcul de Namur.

Je remercie mon frère pour la correction attentive de ce texte.

Enfin, je tiens à dire merci à tous ceux qui m'ont accompagné et soutenu tout au long de mes études et de mon mémoire, en particulier mes parents et la communauté du CRU.

A ma famille.

A Dominique.

INTRODUCTION

Ce travail a pris sa genèse dans le constat du malaise qui règne dans le domaine de l'aide, et en particulier de la documentation, à l'utilisateur. Souvent, quand une question se pose à quelqu'un, le seul moyen d'avoir une réponse est de trouver 'quelqu'un qui sait', et ce malgré l'existence de plusieurs autres sources d'aide à divers endroits.

La première partie essaie de cerner le problème ; elle se base sur une série d'entrevues avec des utilisateurs et des responsables de Centres de Calcul, un aperçu de l'existant, et un examen de la littérature.

La deuxième partie décrit une proposition de solution, dont la partie informatique sera décrite en détail. On décrira la philosophie, l'élaboration et le fonctionnement des programmes créés : AIDE (Assistance Informatique pour utilisateurs Débutants et Expérimentés) et ses logiciels annexes de gestion d'écran GETEDI (Gestion d'Ecran pour TErminaux DIvers) et SCREEN (Système de CRéation et d'Edition d'EcraNs) et de mise en forme de documentation INTRO.

PREAMBULE

Dans l'introduction, nous venons de parler d'"aide" et de "documentation à l'utilisateur". Avant d'aller plus loin, essayons de définir un certain nombre de concepts dont on va traiter.

L'aide à l'utilisateur consiste essentiellement à lui fournir de l'"information" lui permettant d'utiliser efficacement l'ordinateur et ses logiciels. Une des principales sources d'information est la **documentation** sous toutes ses formes.

Il faut resituer le problème de la documentation dans le contexte plus général de l'interface homme/machine. Au sens large, on peut considérer comme faisant partie de cet interface tout ce qui régit l'interaction entre l'homme et la machine (p.ex. ergonomie du poste de travail, etc) ; la documentation sous tous ses aspects en fait partie. Dans un sens plus restreint, c'est tout ce qui, dans un programme, gère la communication des paramètres d'entrée et des résultats. Cette interface doit de préférence être facile et agréable pour l'utilisateur.

Le sujet des interfaces homme/machine (au sens restreint comme au sens large) est un sujet actuellement en pleine évolution, et fort complexe. Il ne sera pas possible de faire le tour du problème dans le cadre de ce travail; mais il est important d'évoquer ne fusse qu'un peu le sujet, car il intéragit avec le problème de la documentation à deux niveaux :

- à Complexité égale, une interface bien conçue peut réduire la grosseur du manuel de documentation de façon non négligeable.
- Les éventuelles solutions informatiques à l'un ou l'autre problème de documentation ne seront utilisables que si nous respectons les règles de base de l'interfaçage homme/machine.

CHAPITRE 1

APERCU DU PROBLEME

1.1. Introduction

- "Je voudrais obtenir des résultats statistiques sur une enquête que je viens de réaliser. Existe-t-il un logiciel que je pourrais employer ? " (un économiste)

- "Va voir X, lui s'y connait en ordinateurs ". Et X répond qu'effectivement il existe un logiciel dont Y est spécialiste...mais Y ne travaille plus aux facultés depuis deux mois...

- ...

- "Où trouver de la documentation sur le package de programmation linéaire LAMPS ? " (un mémorant d'informatique)

- "Si je me souviens bien, il doit y avoir un HELP sur le système...Il doit aussi y avoir un manuel au Centre de Calcul...Si tu ne l'y trouves pas, il y en a peut-être une copie au département de Mathématique.

- ...

- "Dis, je ne me souviens plus quelle est la différence entre ce fichier 'essai1.for' et cet autre, 'essai2.for' " (un mathématicien)

- "Tu peux faire 'compare windows', avec l'éditeur EMACS, ou alors employer SRCCOM, qui est un programme spécialisé ! "

- ...

- "Je suis parti à l'étranger il y a six mois...La version du traitement de texte SCRIBE que j'utilisais a-t-elle changé ? " (un chercheur)

- ...

- "Mais où donc se situe l'interrupteur de ce f... terminal ? " (un étudiant de première candi)

Ces questions et dialogues sont imaginaires, et quelque peu caricaturaux. Mais c'est néanmoins le genre de chose qu'on peut entendre dans les pools de terminaux et les couloirs d'un Centre de Calcul. La réponse à ces questions est souvent fournie par le voisin, un ami, l'assistant , ou éventuellement quelqu'un de ce Centre, mais ces personnes ne sont pas toujours disponibles et/ou compétentes dans le domaine précis de la question. Il faut alors parfois entamer un véritable 'jeu de piste ' pour trouver une réponse.

Pourtant, il existe des milliers de pages de documentation sur le système,

mais on ne sait pas à quel endroit trouver le renseignement. Il y a une bonne probabilité que le programme que l'on cherche ait déjà été écrit par quelqu'un, mais personne ne le sait. Le problème principal est donc celui de l'accès à l'information.

1.2. Aperçu de l'existant

La première chose à faire si l'on veut cerner le problème est de savoir d'où l'on part. On va donc rapidement passer en revue les moyens dont on dispose à Namur.

Les sources principales d'aide et de documentation sont :

1. Les manuels classiques.

   Fournis en un certain nombre d'exemplaires au moment de l'achat d'un software et/ou dupliqués, ils sont en vente au secrétariat du Centre de Calcul (quand ils ne sont pas épuisés). Certains départements en possèdent des copies, à usage interne. Ces manuels sont le plus souvent en anglais.

2. Les fichiers "help".

   Il s'agit en règle générale de fichiers de quelques pages présentant un aspect de l'ordinateur (essentiellement des softwares, mais aussi un horaire particulier, les réservations des pools des terminaux, etc). Ces fichiers sont souvent fait-maison et donc en français. Ils sont accessibles facilement via le système, en tapant simplement HELP 'sujet'. On peut obtenir un catalogue des fichiers HELP par l'intermédiaire de commandes du système d'exploitation. On obtient alors une liste d'environ 200 noms de programmes, sans autre commentaire. Au total, la dimension des fichiers est de 270 pages.

3. Les fichiers "doc"

   Il s'agit ici de fichiers plus volumineux (4000 pages pour 80 fichiers), plutot destinés à être imprimés. (Ils sont parfois formatés pour l'imprimante par traitement de texte) . Ils reprennent souvent un manuel d'utilisation complet, en anglais ou en français. A cause de problèmes d'espace disque, ils ne sont disponibles que pendant la journée. Le seul moyen d'obtenir une liste des fichiers existants est de donner la commande DIR pour la directory concernée.

   Il est à noter qu'il n'y a aucune articulation précise entre les fichiers HELP et les fichiers DOC : pour certains programmes, les deux existent, pour d'autres seulement l'un ou l'autre, ou même rien du tout 'en ligne '.

4. Xinfo

   Il s'agit d'un programme donnant accès a une documentation en

anglais, structurée sous forme d'arbre. La racine de l'arbre est le menu général, présentant les sujets sur lesquels la documentation est disponible. Les embranchements suivants sont des menus intermédiaires permettant d'affiner la sélection du sujet auquel on s'intéresse, et les feuilles contiennent le texte proprement dit. Les commandes principales dont on dispose sont : avancer, reculer dans l'arbre,passer à la feuille suivante ou précédente au même niveau, retourner à la racine, retourner à l'endroit d'où l'on vient. Il existe aussi, quand le sujet le demande, une possibilité de cross-reference d'une feuille à une autre.

Les sujets documentés le sont en général très complètement, mais cette documentation n'est pas tenue à jour, et les sujets documentés sont peu nombreux. De plus, ce ne sont pas toujours des sujets d'intérêt général. (Xinfo est un programme créé par le MIT et tourne donc autour des préoccupations locales)

5. "Messages au système" et BBOARD.

   Les "messages au système" sont de courts messages apparaissant lors du 'login' de tout utilisateur. BBOARD est un logiciel permettant d'afficher des messages sur un 'tableau d'affichage', messages que chacun peut consulter quand il le désire. Ces systèmes sont donc des moyens généraux de communication entre utilisateurs. Ils nous intéressent ici dans la mesure ou ils peuvent servir à faire part de mises-à-jour ou de nouveautés aux utilisateurs.

6. Assistance intégrée aux programmes, quand elle existe.

   Il s'agit parfois juste de la possibilité d'afficher un fichier HELP brut à partir du programme utilisé.Ce cas ne sera pas traité séparément dans ce travail(il suffit de se référer à ce qu'on dira des fichiers HELP et DOC) .

   L'autre cas est plus intéressant : l'aide fournie est dépendante du contexte, afin de répondre à la question que l'utilisateur peut se poser à un moment donné. Cette aide est accessible en tapant 'help', un point d'interrogation, ou encore en appuyant sur une touche spéciale (pour les terminaux ayant cette caractéristique).

7. Aide de personne à personne, quand c'est possible.

1.3. Objectivation du malaise

Afin de mieux cerner le problème, une petite enquête a été menée auprès d'un échantillon d'utilisateurs. De plus, l'expérience d'un stage de quatre mois dans des Centres de Calcul et des discussions avec leurs responsables ont permis de se rendre compte directement du genre de questions que l'on rencontre.

1.3.1. Enquête au Centre de Calcul de Namur

On a sélectionné une dizaine de personnes, de facultés, d'intérêt et de formation très différente. Ces personnes sont :

- Une secrétaire, dont le travail consiste essentiellement en l'utilisation de logiciels d'intérêt général: Scribe, Emacs, MM
- Une personne de l'administration
- Un mathématicien-informaticien,qui est responsable des moyens informatiques au niveau du département de mathématiques (applications habituelles: beaucoup d'utilisation de librairies et de programmes provenant tant de l'extérieur que de la "maison").
- Un assistant et doctorant en sciences économiques
- deux assistants en biologie,qui s'occupent essentiellement de recherche; ils utilisent des logiciels spécialisés et autres (Emacs, Scribe, SPSS, BMDP,...) et programment en divers langages : Basic, Fortran, APL,...

  Ils s'occupent aussi de l'encadrement de "naïfs": chercheurs, mémorants, doctorants, étudiants, ayant besoin sporadiquement de l'ordinateur.
- Un chef de travaux en médecine.
- Une dactylo, novice en informatique, engagée comme C.S.T. pour l'introduction de données pour une base de données juridiques; elle utilise essentiellement Emacs.
- Un premier assistant en informatique.
- Un chercheur en informatique.

Ces personnes ont été contactées pour une entrevue, dirigée par le schéma ci-dessous :

- Présentation générale du travail et des centres d'intérêt de la personne.
- Comment l'initiation à l'informatique a-t-elle été faite et comment se fait-elle ?
- Avis sur les systèmes de documentation, sur d'autres systèmes pouvant servir à communiquer des modifications, etc.
- Manques dans le domaine de l'aide, souhaits,...

Ces interviews ont permis aux utilisateurs interrogés d'évoquer de nombreux problèmes et de suggérer parfois l'une ou l'autre solution.

On trouvera le rapport complet de chaque entrevue en annexe. En synthèse, voici quelques tendances qui se dégagent :

- Plusieurs personnes interrogées ont soulevé le problème de l'initiation. Aucune structure n'existe pour les nouveaux utilisateurs de l'informatique (e.a., il n'y a pas de "manuel du débutant"). La formation est laissée à la complète initiative du professeur ou de l'assistant dans le cas d'étudiants, ou d'une personne de bonne volonté si le novice n'est pas intégré dans un environnement d'apprentissage. Le même problème se pose lorsque quelqu'un désire utiliser un nouveau logiciel; la seule solution est souvent d'ingurgiter d'un seul coup le manuel complet...

- Les problèmes principaux de la documentation telle qu'elle existe concernent l'accessibilité et la mise à jour. Les problèmes d'accessibilité sont dus au désordre dans lequel elle se trouve et de l'impossibilité de savoir de manière simple et efficace ce qui existe. La langue anglaise utilisée dans la majorité des fichiers d'aide, la difficulté de maîtriser certains programmes de documentation et d'avoir une trace écrite de ce qui apparaît posent des problèmes supplémentaires. Au niveau de la MAJ, pour peu que l'on ait manqué le message au système annonçant une nouvelle version d'un logiciel, on ne se rend compte qu'il y a eu un changement qu'au moment où une application qui a toujours tourné "se plante" sans raison apparente...Il arrive aussi que, confronté à un problème, on soit tout heureux d'en trouver la solution...fausse, dans un manuel qui n'est plus à jour...

- Enfin, beaucoup insistent sur l'importance des contacts de personne à personne dans la résolution de problèmes. Certains aimeraient avoir une liste formelle de personnes 'consultables' en cas de problème avec un logiciel particulier. On aimerait aussi avoir de temps à autre un séminaire de présentation des nouveautés logicielles...Par contre, l'idée d'un appel à l'entraide publique (sous forme, p.ex. d'un "message à tous" posant une question) n'est jamais envisagée et suscite des réticences quand elle est proposée (il y a probalement une sorte de pudeur de son "ignorance" vis-à-vis des autres utilisateurs).

1.3.2. Point de vue de Centres de Calcul

L'essentiel des renseignements que nous avons obtenus proviennent du Centre de Calcul de Namur; mais notre court séjour dans deux autres centres (au Centre Mondial pour l'Informatique et les Ressources Humaines, de Paris, et chez la firme Schlumberger, à Paris également) a montré que le même genre de questions se posent ailleurs.

En ce qui concerne le Centre de Namur, les principaux types de questions qui reviennent peuvent être classés comme suit :

- Une bonne partie des questions sont des demandes d'aide pour la

détection d'erreurs dans un programme (souvent Fortran) ou un logiciel connu de l'utilisateur. Parmi ces erreurs, on compte 10% d'erreurs 'système ', 70% de "bêtes" erreurs de programmation et 20% d'erreurs liées à une méconnaissance d'un point particulier du logiciel utilisé.

- Une autre partie des demandes sont des demandes d'aide pour l'utilisation d'un package qui est disponible sur le système.

- Enfin, la troisième catégorie de demandes sont des questions plus générales : les gens ont un problème qu'ils espèrent pouvoir résoudre grâce à l'informatique.

1.3.3. Conclusion

En résumé, les besoins des utilisateurs peuvent se classer en trois catégories :

1. **Formation** à l'informatique en général ou à un logiciel
2. **Information** quant à l'existant
3. **Référence** en cas de question et de problème

1.4. Eventail des systèmes d'aide possibles

Cet éventail reprend les systèmes présents ici à Namur et divers systèmes présentés dans la littérature de ces dernières années. [1, 2, 3, 4, 5]Pour structurer cette liste, nous avons adopté une classification par **medium**.

- Ecran.

  * XINFO: voir description supra.

  * Programmes d'apprentissage en-ligne ('On-line tutors') . Ces programmes sont des 'visites guidées' des capacités d'un logiciel. L'utilisateur peut s'exercer aux différentes commandes en même temps qu'est affiché le texte concernant celles-ci.

  * Aide en-ligne incluse dans le programme. voir supra.

- Ecran avec possibilité de copie papier.

  * Fichiers HELP et DOC. Voir supra.

  * KWIC index. [4] Il s'agit d'un système d'index permettant un accès pseudo-intelligent à un simple fichier de texte; on fournit un mot-clé, et on obtient en retour une fenêtre de taille fixe entourant une (première) occurence du mot dans le fichier.

  * DOCUMENT. [4] DOCUMENT est un logiciel de documentation très

original développé aux Etats-Unis, à Livermore, dans le cadre du National Magnetic Fusion Energy Computer Center (NMFECC). Le NMFECC est un organe fédéral qui dispose d'un réseau comportant un CDC7600 et deux CRAY-1; le développement des logiciels se fait à Livermore, mais les utilisateurs sont répartis sur l'ensemble du territoire des Etats-Unis. De ce fait, les méthodes classiques de documentation (manuels fournis à une source) sont inapplicables. DOCUMENT permet d'accéder sur une base **sémantique** à la partie de la documentation qui intéresse l'utilisateur, et d'afficher ou d'imprimer localement cette partie. DOCUMENT fournit aussi des services annexes : liste de mots-clés pour chaque fichier d'aide, liste des ajouts de documentation et des maj par ordre chronologique, et liste alphabétique et par thème des logiciels existants. Pour être accessible à DOCUMENT, le texte de la documentation doit être divisé en fenêtres entourées ("parenthèsées") par des mots-clés, ces mots-clés indiquant la nature de la documentation qu'ils entourent; plusieurs fenêtres peuvent s'imbriquer. Exemple :

```
------------------------------ ENTIER
----------------------INTRO
---------------QUOI

fonction du logiciel

---------------QUOI
---------------QUI

pour tout renseignement...

---------------QUI
-----------------------INTRO

------------------------ARGUMENTS
-------IN

        -----BIZARRERIE

-------IN
-------OUT

        -----BIZARRERIE

-------OUT
------------------------ARGUMENTS
------------------------------ ENTIER
```

Lors de la création des programmes, DOCUMENT fournit un utilitaire facilitant la mise en forme de la documentation.

* Messagerie électronique. voir supra.

- Papier.

* Manuels. Il peut s'agir de manuels de base ou approfondis.

* Cartes de référence. Après la période d'apprentissage, les gros

manuels sont le plus souvent abandonnés au profit d'aides-mémoire soit 'bricolés' par l'utilisateur (ce qu'on a appelé "cheat sheet" [5]), soit fournis par le concepteur du logiciel sous forme de 'reference card'.

- Contact direct.

  * Cours, donné par quelqu'un du Centre de Calcul, par le responsable d'un groupe d'utilisateurs ou une personne spécialiste dans un domaine particulier.

  * Service d'aide institutionnelle du CC. [2, 3] Une personne est désignée définitivement ou par roulement pour répondre aux questions des utilisateurs.

  * Entraide. Entre utilisateurs se connaissant.

1.5. Evaluation

Cette évaluation s'inspire de celle proposée dans [4]

Un système de documentation doit apporter une réponse à quatre types de problèmes :

- Problème d'accès : quelles sont les possibilités de trouver la réponse à une question que l'on se pose ?
- Problème de publication : quelles sont les possibilités de 'publication' (= d'avoir une trace écrite) des informations reçues ?
- Problème d'administration : les moyens d'accès et de publication choisis sont-ils praticables du point de vue du coût, des maj, etc ?
- Problème de la qualité : les moyens utilisés pour répondre aux problèmes d'accès, de publication et d'administration sont-ils une aide ou une difficulté dans l'accomplissement de la mission de tout système de documentation, càd la fourniture d'informations lisibles,structurées et à jour ?

Les divers systèmes de documentation fournissent une réponse plus ou moins complète à ces problèmes; de plus, il faut se demander comment cette réponse est fournie...Pour cette évaluation, on distingue quatre 'critères de mérite'

- simplicité : le système s'emploie-t-il de façon naturelle et sans complication inutile ?
- clarté : le système aide-t-il l'utilisateur à employer ses caractéristiques propres ? (système auto-documenté, avec help, etc)

- flexibilité : le système est-il tolérant vis-à-vis des erreurs de l'utilisateur ? Lui fournit-il ce dont il a besoin ?

- contrôle de l'utilisateur : dans quelle mesure les actions du système sont-elles contrôlées par l'utilisateur plutôt qu'imposées par des contraintes du système ?

1. Xinfo :

accès : en ligne; la structure d'arbre impose parfois un chemin assez long avant d'arriver à la feuille concernant le sujet cherché. (Ce chemin peut être court-circuité dans le cas (assez rare) où l'on connait le nom exact du noeud cherché)

publication : pas prévue, difficile, de mauvaise qualité.

administration : obligation d'écrire (de réécrire) les fichiers de documentation en les structurant dans le format arborescent de Xinfo (il existe un set d'outils facilitant ce travail)

qualité : avantages et inconvénients de la structure d'arbre : bonne documentation si on y accède séquentiellement.

Critères de mérite :

- simple : non!, selon l'avis d'une majorité d'utilisateurs
- clair : système auto-documenté, pas toujours très facile à aborder (pas de manuel papier !!)
- flexible : oui (on peut visiter certains noeuds seulement)
- contrôle de l'utilisateur : total, quand on a appris à utiliser les commandes

2. Programmes d'apprentissage en-ligne

accès : en ligne ; le plus souvent, accès à un nombre limité de concepts(pour des raisons pédagogiques).Dans certains cas, tout le logiciel est documenté de cette façon.

publication : pas prévue...sauf si on fait imprimer le texte de tout le cours

administration : très lourde : le cours doit être créé sur mesure pour chaque logiciel...

qualité : par nature, le système doit fournir (s'il est bien conçu) une formation bien adaptée pour les débutants

Critères de mérite :

- simple,clair : oui (par nature)

- flexible : peu flexible : il faut suivre la progression prévue par le concepteur du cours

- contrôle de l'utilisateur : faible (à part quitter le cours)

3. Aide intégrée au programme

accès : en ligne ; si l'aide est dépendante du contexte, il y a de bonnes chances de trouver la réponse à sa question.

publication : pas prévue et inutile dans ce contexte.

administration : ce type d'aide doit être prévu dès la conception du programme. Rajouter cette aide par après est un travail très lourd.

qualité : par nature, le système doit fournir (s'il est bien conçu) une aide pratique et adaptée.

Critères de mérite :

- simple,clair : en général oui, surtout si l'appel de l'aide se fait d'une façon standard sur tous les logiciels d'un site.

- flexible : oui, l'utilisateur ne reçoit de l'aide que quand il le demande, sur le sujet qu'il demande.

- contrôle de l'utilisateur : complet (car sur demande)

4. Fichiers HELP et DOC :

accès : online et offline, brut (pas de possibilité de sélection de la partie accédée)

publication : brut, mais sur demande, donc a priori informations à jour

administration : aucune facilité particulière (en particulier, pas de lien entre Help et Doc pour les maj)

qualité : avec un fichier brut, il est difficile de contenter tant les débutants que les utilisateurs chevronnés

Critères de mérite :

- simple : oui (on ne peut plus simple)

- clair : sans objet

- flexible : non

- contrôle de l'utilisateur : faible (à part "^C")

5. KWIC index :

accès : en ligne ; la qualité et la précision de l'accès dépendent du type d'information recherché et du mot-clé choisi.

publication : eventuellement, hardcopy (peu utile pour l'usage prévu)

administration : le système peut fonctionner sur un matériau de base classique : fichier texte (pas d'effort supplémentaire de mise en oeuvre)

qualité : il est peut-être préférable de structurer la documentation en ayant à l'esprit le mode de fonctionnement du "KWIC index".

Critères de mérite :

- simple et clair : par nature, ce système est simple (certains diront "rudimentaire...")
- flexible : non (on obtient une fenêtre fixe autour du mot clé)
- contrôle de l'utilisateur : moyen (on obtient le contexte autour de la première occurence du mot recherché)

6. DOCUMENT :

accès : en ligne, accès par thème et mots-clés; la structure de mots-clés choisie permet de n'accéder qu'à la partie de documentation cherchée et avec un degré de détail contrôlable.

publication : souple, à la demande, les parties sélectionnées seulement

administration : facilitée par un certain nombre d'outils; ce système exige néanmoins une refonte "intelligente" de la documentation pour avoir toute sa puissance

qualité : bonne si le documenteur est bon...(tout est fait du point de vue forme; il faut que le fond suive)

Critères de mérite :

- simple,clair (autodocumenté),flexible et controlable à la fois!

7. Courrier électronique, Bboard

Il s'agit ici de systèmes dont le but est plus général que celui des systèmes de documentation. Dans le cadre présent, on en fait une utilisation très particulière et limitée : ils sont surtout employés pour annoncer de nouveaux logiciels ou des changements dans ceux qui existent déjà. (On pourrait aussi les utiliser dans des buts d'entraide, mais ceci n'est pas entré dans les moeurs).

8. Manuel (de base ou approfondi)

accès : brut + index éventuel

publication : stock de manuels constitué à l'avance, dans lequel on puise à la demande (!!!attention à la péremption des manuels...)

administration : procédé assez coûteux pour des manuels classiques (composition, impression, reliure, etc). De plus, il y a un problème en ce qui concerne les mises-à-jour.

qualité : peut être très bon ou très mauvais, selon le cas (avec les réserves faites ci-dessus concernant les mises-à-jour) . Le problème majeur lié à la nature même du médium est qu'un manuel est destiné à un public, qui est censé avoir un certain background en informatique. Les gens expérimentés trouveront le manuel trop détaillé, d'autres, novices, le trouveront trop compliqué.

Critères de mérite :

- simple : oui
- clair : sans objet
- flexible : non
- contrôle de l'utilisateur : possible (il consiste à fermer le manuel ou tourner la page...)

9. Carte de référence

Ces aide-mémoire doivent accompagner un livre ou un cours et sont insuffisants seuls.

10. Cours

accès : dépend de l'organisation...a priori, facile.

publication : il faut prévoir des notes à distribuer en support au cours (+ notes personnelles des élèves, reprenant seulement ce qui les intéresse)

administration : assez lourd

qualité : dépend de l'enseignant

Critères de mérite :

- simple,clair : oui, car les personnes sont habituées à suivre les cours
- flexible : peu de flexibilité à l'intérieur d'un cours (on peut prévoir plusieurs types de cours pour répondre à plusieurs types d'attentes)

- contrôle de l'utilisateur : peu de contrôle sur l'information reçue (si ce n'est de 'brosser' les cours...)

11. Service d'aide, conseillers

accès : dépend de l'organisation du service

publication : consiste en les notes manuscrites résultant de l'entretien.

administration : assez lourd; coûteux s'il faut engager du personnel complémentaire.

qualité : dépend de la compétence technique et pédagogique de l'advisor.

Critères de mérite :

- c'est un système auquel l'utilisateur recourt de façon naturelle :il est simple, clair, flexible et contrôlable par l'utilisateur...

12. Entraide

Quand il fonctionne bien, ce système est très intéressant : il regroupe les avantages d'un service d'aide, sans ses coûts. Le principal problème (de taille) est que la qualité de la réponse fournie dépend fort de la personne auprès de laquelle on s'est renseigné...

1.6. Adéquation entre les besoins et les solutions possibles

On a passé en revue les problèmes de documentation qui se posaient (formation, information, référence) ainsi que les moyens de documentation existants. On va maintenant établir la correspondance entre les besoins et les types de réponses possibles.

1. Problèmes de **formation** de base pour 'débutants' : initiation à l'exec et aux commandes de base pour nouveaux utilisateurs, et de formation aux bases d'un package particulier, pour nouveaux utilisateurs de ce package.

   - Cours
   - Aide par un 'gourou' local
   - Manuel de base, accompagné d'une feuille de référence rapide
   - Programmes d'apprentissage en-ligne

2. Problèmes d'**information** sur l'existant

- répertoire de programmes par thèmes (avec chaque fois un court descriptif).

- répertoire des nouveautés achetées ou "faites-maison".

- Bboard, pour annoncer les nouveautés.

Problèmes d'**information** du genre "existe-t-il quelque chose pour résoudre tel problème ?"

- Répertoire de programmes par thème

- Contact direct avec le centre de calcul, un service d'aide spécialisé,...

Problèmes d'**information** quant aux mises-à-jour :

- Bboard, pour afficher les modifications prévues

- Courrier électronique

- Manuel,fichier de documentation,...mis à jour

3. Problèmes de **référence** (en cas de doute et/ou d'erreur).

- Contact direct avec le centre de calcul ou un service d'aide

- Manuel complet et à jour

- Système de documentation en ligne avec moyen d'accès sélectif (par sujet)

- Entraide (par un utilisateur local expérimenté)

1.7. Fournisseurs de la documentation

La première source de documentation, qui devrait toujours exister, est le fournisseur du programme : actuellement, tout informaticien sérieux reconnaît l'importance d'une bonne documentation à tous les stades du développement d'une application ; idéalement, au stade de la distribution, la majeure partie du travail devrait être faite...Le problème est que, par manque de temps, d'argent, par paresse, ou encore parce que le travail de documentation est moins valorisant que le travail de développement, cette étape du travail est négligée et la documentation est souvent embryonnaire si pas totalement inexistante...

Aussi, les services (plus ou moins institutionnels) d'aide des centres de calcul doivent combler les trous ou refondre certaines parties de documen-

tation ; mais ces services sont confrontés aux mêmes types de problèmes que ci-dessus : manque de temps, découragement devant l'ampleur de la tâche, sentiment de faire un travail moins utile que le développement d'un projet, car on ne voit pas directement l'utilisateur qui va profiter de l'aide...

1.8. Interface h-m

Les problèmes d'interface humaine [6, 7, 8, 9] interviennent dans ce travail à deux niveaux :

- Une interface bien conçue peut réduire (ou même, selon certains, supprimer) la nécessité d'autres types d'aide dans l'apprentissage et l'utilisation d'un programme. En poussant ce raisonnement jusqu'au bout, on en arrive à produire des logiciels sans documentation utilisateur... Néanmoins, des expériences ont montré [1]qu'un apprentissage basé uniquement sur des moyens 'en-ligne' est moins efficace que s'il y a un support d'une autre nature en sus.

- A partir du moment où l'on a décidé de fournir un outil de documentation informatisé, ce qu'on a dit au paragraphe précédent doit être appliqué et on doit donc veiller à concevoir une interface homme/machine ergonomique.

Les principes de base à garder à l'esprit quand on conçoit une interface humaine sont[1] :

- l'interface humaine doit être basée sur un modèle conceptuel familier à l'utilisateur
- voir et pointer sont meilleurs que retenir et taper au clavier
- les commandes doivent être cohérentes et précises
- les choses simples doivent être simples et les choses compliquées possibles
- un help (éventuellement avec exemples) doit être toujours disponible

La façon dont ces principes pourront être appliqués dépend entre autres de la technologie de l'instrument de communication homme/machine disponible. Les moyens les plus couramment rencontrés sont (dans l'ordre de facilité croissante) :

---

[1] ces renseignements proviennent essentiellement d'entrevues avec des informaticiens lors du séjour à Paris

- imprimante ou écran en mode scroll
- écran avec gestion d'écran
- écran graphique
- écran graphique interactif (souris, crayon lumineux, ...)

Dans la plupart des configurations, le medium le plus répandu est, au mieux l'écran avec gestion d'écran. Dans ce cas particulier, nous pouvons citer quelques principes supplémentaires :

- la page doit être subject-oriented
- un help doit être disponible à tout moment, sous forme d'un help en une ligne, préservant l'écran, et d'un help plus complet pleine page.
- contrôle d'erreur d'introduction dès que possible
- diagnostic d'erreur aussi explicite que possible (complet,proposant si possible un remède)
- signaler quelque part sur l'écran qu'un long processus est en cours

1.9. Conclusions

Dans la recherche d'une bonne utilisation de l'outil informatique, on se rend de plus en plus compte de l'importance d'un bon interfaçage homme/machine; les problèmes de documentation sont donc aussi mis en exergue.

Souvent, les sources de documentation existent, sont même abondantes; mais, faute de structure interne et de méthode d'accès facile, elles exigent un effort trop important pour être utilisées.

La deuxième partie de ce travail s'attachera à proposer quelques solutions à certains de ces problèmes.

CHAPITRE 2

AIDE ET LES LOGICIELS ASSOCIéS

2.1. Introduction

Nous avons vu dans la première partie les problèmes qui se posaient dans le domaine de la documentation et les divers types de solutions possibles.

Rappelons que les besoins rencontrés sont dans le domaine de la formation, de l'information et de la référence.

Les améliorations que l'on pourrait apporter à la situation existante sont de deux ordres : meilleure organisation dans l'utilisation d'outils existants, et création de nouveaux outils (e.a. informatiques).

Quel genre de solution peut-on proposer, en gardant à l'esprit les contraintes liées à la petite taille de l'université de Namur ?

- Au niveau de l'organisation :

  * Bboard et mail
  * Création d'une bibliothèque de manuels à jour
  * Service de conseillers
  * ...

  La conception et la mise en oeuvre de ce genre de solution sortent du cadre de ce travail.

- Au niveau des outils informatiques :

  Il s'agit de la création d'un logiciel de documentation interactif en ligne.

  Dans le domaine de la formation, l'apport d'un système informatique est marginal; selon certaines études, un tel système est même relativement inefficace [1]. De plus, la conception d'un tel "tuteur en ligne" n'est possible que pour certaines catégories de logiciels et demande un très grand effort.

  Dans les deux autres domaines, par contre, (information et référence), il est possible de créer un logiciel facilitant le travail des utilisateurs en les aidant à trouver la documentation dont ils ont besoin. Parmi les fonctions que pourrait fournir un tel logiciel, nous avons choisi les suivantes :

* Information par rapport à l'existant :

  - liste des noms des programmes par ordre alphabétique

  - accès aux programmes par mot-clé

* Fourniture d'une liste des changements (modifications et nouveautés) depuis une date donnée

Ce logiciel sera destiné à un public qui varie depuis l'étudiant de biologie, néophyte parfait, jusqu'à l'informaticien chevronné. L'interface utilisateur devra prévoir une aide pour le premier, sans être une gêne pour le second. Pour rendre l'interaction homme-machine aussi agréable que possible, on utilisera la gestion d'écran.

Ce logiciel devra éventuellement être adapté sur d'autres ordinateurs de l'université.

2.2. Description des logiciels

Nous avons été conduits à réaliser deux logiciels :

1. AIDE (Assistance Informatique pour utilisateurs Débutants et Expérimentés) : le lecteur de documentation proprement dit. Il est accompagné du logiciel INTRO, qui permet de mettre en forme les informations pour AIDE

2. GETEDI (Gestionnaire d'Ecran pour TErminaux DIvers) : Un gestionnaire d'écran créé pour supporter l'interface homme-machine d'AIDE. Au sein de GETEDI, la gestion physique des différents terminaux est prise en charge par TSCR. TSCR puise ses données dans un fichier créé par MKDESC (Ces deux derniers programmes ont été écrits par Michel DEBAR, du Centre de Calcul des Facultés Universitaires de Namur). Ce gestionnaire est accompagné par SCREEN (Système de CRéation et d'Edition d'EcraN), qui est un mini-éditeur permettant de créer des écrans pour GETEDI.

Nous allons maintenant passer en revue les différents programmes et en donner la structure générale (les critères de structuration employés seront explicités après cette description, afin de pouvoir les illustrer d'exemples).

2.2.1. AIDE

2.2.1.1. Description fonctionnelle

AIDE est un programme interactif d'accès à des "fiches signalétiques" de programmes.

En entrée, il accepte une suite de commandes ayant la forme suivante :

```
[suite commandes] ::= [commandes] QUITTER
[commandes] ::= [commandes] [commande]
[commande] ::= NOM [nom de programme]
               or DATE [date]
               or MOT-CLE [liste de mots-clés]
[liste de mots-clés] ::= [mot-clé]
                 or [mot-clé] [séparateur] [liste de mot-clés]
[séparateur] ::= { , or ; or . or : }
```

Un [nom de programme] est une chaîne de caractères (max. 20). Une [date] est constituée d'une chaîne de caractères respectant la contrainte JOUR-MOIS-ANNEE, le mois pouvant être en toutes lettres (en anglais ou en français) ou sous forme numérique, l'année pouvant avoir 2 ou 4 chiffres (19XX ou XX). Un [mot-clé] est une chaîne de caractères (max.20).

Une fiche signalétique comporte au minimum les renseignements suivants :

- Nom du programme (20 car.)
- Fonction (80 car.)
- Date de màj
- Sources de documentation (3 lignes de 80 caractères)
- Mots-clés : 10 mots de 20 car.

Une commande incorrecte provoque l'affichage d'un message ad hoc.

L'utilisateur dispose d'une possibilité d'aide chaque fois qu'il doit introduire quelque chose.

La sémantique des commandes est la suivante :

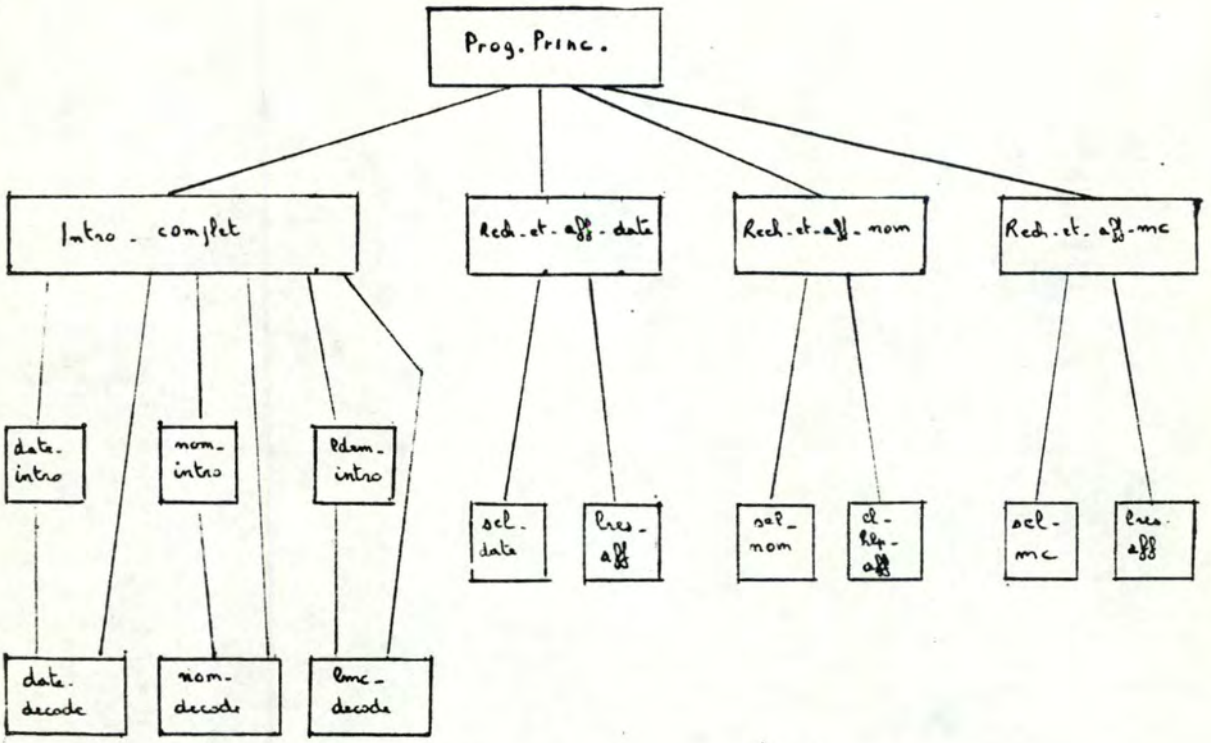
NOM [nom de programme] : la fiche signalétique du programme de nom [nom de programme] est affichée.

DATE [date] : on sélectionne tous les programmes dont la fiche a été créée ou modifiée après la date [date] et on en affiche le nom et une courte description extraite de la fiche signalétique.

MOT-CLE [liste de mots-clés] : on sélectionne tous les programmes qui ont parmi leurs mots-clés les mots-clés de la liste. La comparaison ne doit pas se faire sur une base d'égalité stricte, mais doit tolérer de petits écarts dans la façon d'orthographier le mot-clé. L'utilisateur doit avoir un moyen d'affiner sa sélection.

Contraintes particulières d'implémentation : on veillera tout particulièment à la qualité de l'interface utilisateur (utiliser la gestion d'écran)

2.2.1.2. Structure des modules



Procédure intro_complet;

- IN : introduction au terminal

- OUT : com_cour : commande sous forme interne

  code_ret : code retour

- Sémantique : Lit une chaîne de caractères à partir du terminal
  si elle représente une commande correcte, garnit la commande "com_cour" avec la représentation sous forme interne de cette commande;
  sinon, le code retour indique la raison de l'échec.

  Gère aussi la fourniture de l'aide.

Procédure date_intro;

- IN : introduction au terminal

- OUT : str : chaîne de caractères

  code_ret : code retour

- Sémantique : garnit la chaîne "str" avec une date lue au terminal
  Le code retour indique la condition de terminaison.

  Gère aussi la fourniture de l'aide.

Procédure nom_intro;

- IN : introduction au terminal

- OUT : str : chaîne de caractères

code_ret : code retour

- Sémantique : garnit la chaîne "str" avec un nom de programme lu au terminal; le code retour indique la condition de terminaison.

Gère aussi la fourniture de l'aide.

Procédure ldem_intro;

- IN : introduction au terminal

- OUT : com_cour : commande sous forme interne

code_ret : code retour

- Sémantique : garnit la chaîne "str" avec un ou des mot(s)-clé(s) lu(s) au terminal;
le code retour indique la condition de terminaison.

Gère aussi la fourniture de l'aide.

Procédure date_decode;

- IN : str : chaîne de caractères

- OUT : cde : commande sous forme interne

cod_ret : code retour

- Sémantique : Essaie de transformer "str" en date sous forme interne. En cas de réussite, garnit la partie "date" de "cde" avec cette date sous forme interne et met le code retour à 'ok'.
Si "str" ne représente pas une date correcte, le code retour indique l'erreur détectée.

Procédure nom_decode;

- IN : str : chaîne de caractères

- OUT : cde : commande sous forme interne

cod_ret : code retour

- Sémantique : Garnit la partie "nom" de "cde" avec le premier mot de la chaîne de caractères "str".
Le code retour indique le cas ou "str" est vide ou vaut 'quitter'.

Procédure lmc_decode;

- IN : str : chaîne de caractères

- OUT : cde : commande sous forme interne

cod_ret : code retour

- Sémantique : Transforme les mots de la chaîne de caractères "str" en une liste de mots-clés, qui sera désignée par la partie "ldem" de "cde".
Le code retour indique le cas ou "str" est vide ou vaut 'quitter'.

Procédure rech_et_aff_date;

- IN : cde : commande sous forme interne

- Paramètre global : collection des fiches signalétiques

- Sémantique : Cette procédure crée la liste des programmes dont la fiche signalétique a été mise à jour après la partie "date" de "cde" (sous-module "l_date_sup_search"), et affiche les noms et descriptions des divers programmes (sous-module "lres_aff").

Procédure rech_et_aff_nom;

- IN : cde : commande sous forme interne;

- Paramètre global : collection des fiches signalétiques

- Sémantique : Cette procédure recherche la fiche signalétique correspondant au programme dont le nom est dans la partie "nom" de "cde" (sous-module "lnom_search") et l'affiche (sous-module "el_hlp_aff").

Procédure rech_et_aff_mc;

- IN : cde : commande sous forme interne

- Paramètre global : collection des fiches signalétiques.

- Sémantique : Cette procédure crée une liste des programmes dont la fiche signalétique qui comprend les mots-clés contenus dans la liste désignée par la partie "ldem" de "cde" (sous-module "lmc"). Les

noms et descriptions de ces programmes sont affichés (sous-module "lres-aff"). Il est possible d'affiner la recherche en introduisant des mots-clés supplémentaires.

2.2.1.3. Format des interfaces

Commande sous forme interne : deux champs :

- le premier indiquant le type de commande : 'N' pour NOM, 'D' pour DATE ou 'M' pour mot-clé
- le deuxième dépendant de la valeur du premier :
  * si le premier champ vaut 'N' : chaîne de caractères (max. 20)
  * si le premier champ vaut 'D' : date sous forme interne (AAAA/MM/JJ)
  * si le premier champ vaut 'M' : désignateur vers la représentation d'une liste de mots-clés

2.2.1.4. Composition des collections

Collection des fiches signalétiques : Composition d'un article :

- Nom du programme (20 car.)
- Fonction (80 car.)
- Date de màj : date sous format interne
- Sources de documentation (3 lignes de 80 caractères)
- Mots-clés : tableau de 10 chaînes de 20 car.

2.2.2. INTRO

2.2.2.1. Description fonctionnelle

INTRO est un programme qui permet la gestion des fiches signalétiques de programmes.

A l'entrée, il accepte les commandes suivantes :

```
[suite commandes] ::= [commandes] QUITTER
[commandes] ::= [commandes] [commande]
[commande] ::= AJOUT
               or MISE-A-JOUR
               or SUPPRESSION
```

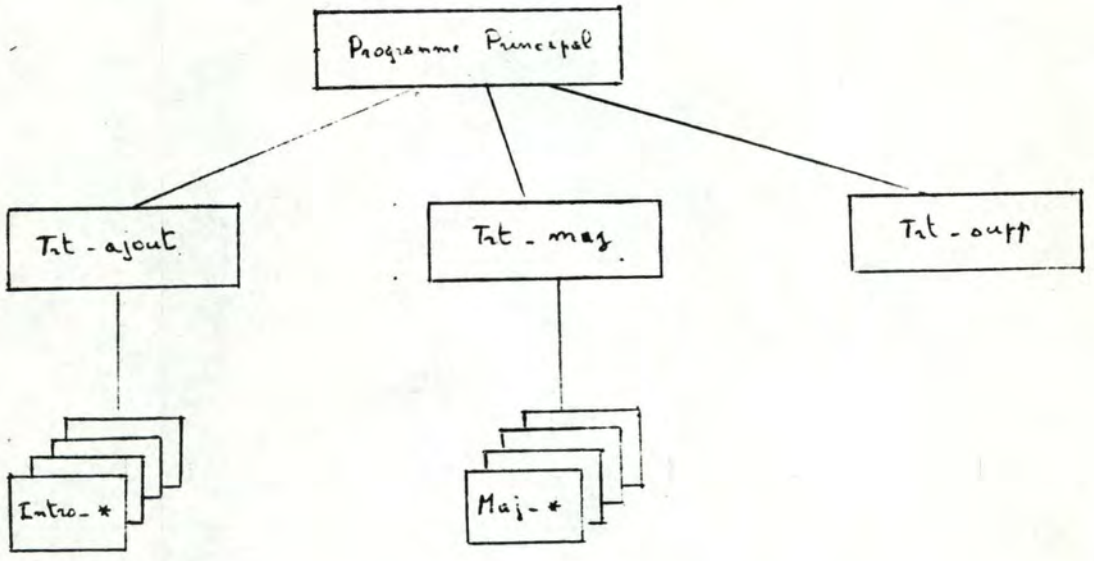
La sémantique de ces commandes est la suivante :

AJOUT : Le programme demande à l'utilisateur d'introduire les champs d'une nouvelle fiche signalétique. On vérifie s'il n'existe pas déjà une fiche au nom du programme. Si c'est le cas, on le signale et on annule l'opération.

MISE-A-JOUR : L'utilisateur peut modifier les champs d'un programme dont il donne le nom. Un défaut est fourni : la valeur ancienne du champ. Si on tente de modifier une fiche inexistante, le programme le signale et annule l'opération.

SUPPRESSION : Le programme supprime la fiche correspondant au programme dont l'utilisateur donne le nom. Si elle n'existe pas, on le signale et on annule l'opération.

2.2.2.2. Structure des modules



Procédures trt_ajout, trt_maj;

- Sémantique : jouent le rôle de coordinateur pour les fonctions intro_* et maj_*

Procédures trt_supp

- Sémantique : demande le nom du programme dont il faut supprimer la fiche; affiche cette fiche; si on confirme l'effacement, supprime la fiche.

Procédure intro_nom;

- IN : entrée au terminal

- OUT : booléen

- Paramètre global : collection des fiches signalétiques

- Sémantique : Si le nom introduit au terminal n'existe pas encore dans la collection des fiches signalétiques, cette procédure garnit la partie "nom" de l'élément courant de la collection des fiches avec celui-ci et "existe" est mis à "false"
sinon, "existe" est mis à "true".

Procédure intro_fonction;

- IN : entrée au terminal

- Paramètre global : collection des fiches signalétiques

- Sémantique : la partie "fonction" de l'élément courant de la collection des fiches signalétiques est garnie avec la chaîne de caractères lue au terminal

Procédure intro_date;

- IN : entrée au terminal

- Paramètre global : collection des fiches signalétiques

- Sémantique : analyse la chaîne de caractères introduite. Tant qu'elle ne correspond pas à une date correcte, redemande l'introduction; la convertit sous forme interne; ensuite, garnit la partie "date" de l'élément courant de la collection des fiches signalétiques avec la date sous forme interne.

Procédure intro_mc;

- IN : entrée au terminal

- Paramètre global : collection des fiches signalétiques

- Sémantique : lit des mots (20 car.) au terminal tant qu'on n'a pas introduit un mot composé de blancs;
Pour chaque mot, vérifie s'il ne s'agit pas d'un mot similaire à un mot-clé existant (2 mots différents, orthographiés différemment). Si c'est le cas, demande s'il faut les considérer comme égaux; ensuite, le mot-clé est mis dans la partie "mot-clés" de l'élément courant de la collection de fiches signalétiques.

Procédure intro_doc;

- IN : entrée au terminal

- Paramètre global : collection des fiches signalétiques

- Sémantique : lit des chaînes de caractères (80 car.) au terminal (max.3) et en garnit la partie "documentation" de l'élément courant de la collection des fiches signalétiques

Procédures maj_*

- Sémantique : Ces procédures ont une fonction comparable aux procédures intro_*; la différence est que lors de la mise-à-jour, on dispose d'une valeur par défaut (valeur ancienne)

2.2.3. GETEDI

2.2.3.1. Description fonctionnelle

GETEDI est un gestionnaire d'écran universel, càd pouvant être adapté à tout terminal. Il traite des fenêtres : représentation de l'aspect que peut avoir la totalité ou une partie de l'écran d'un terminal vidéo, à un moment donné.

Les fonctions minimales à fournir sont les suivantes :

- Affichage de fenêtres
- Introduction dans un cadre de gestion d'écran
- Création, modifications, suppression de fenêtres
- Sauvetage des fenêtres

Les modifications à une fenêtre n'influencent pas directement l'affichage. Les modifications n'apparaissent qu'après avoir donné l'ordre "refresh". L'effet de cette commande est de synchroniser la partie de l'écran décrite par la fenêtre donnée en argument avec celle-ci. Cette façon de faire permet d'avoir plusieurs écrans ou fenêtres en parallèle, d'y faire diverses modifications, puis d'afficher la fenêtre qui convient au moment voulu.

2.2.3.2. Structure des modules

La structure de ce programme est très simple : il s'agit de modules indépendants, constituant une librairie.

2.2.4. SCREEN

2.2.4.1. Description fonctionnelle

SCREEN est un mini-éditeur qui permet de créer, modifier ou supprimer des fenêtres qui seront utilisées par GETEDI.

En entrée, on a les commandes suivantes :

```
[suite commandes] ::= [commandes] QUITTER
[commandes] ::= [commandes] [commande]
[commande] ::= CREATION
               or MODIFICATION
               or EFFACAGE
```

Les opérations d'édition proprement dite permettent les fonctions suivantes :

- Mouvements du curseur
- Ajout de champs
- Suppression de champs
- Rafraîchissement de l'écran
- Tracé d'un cadre

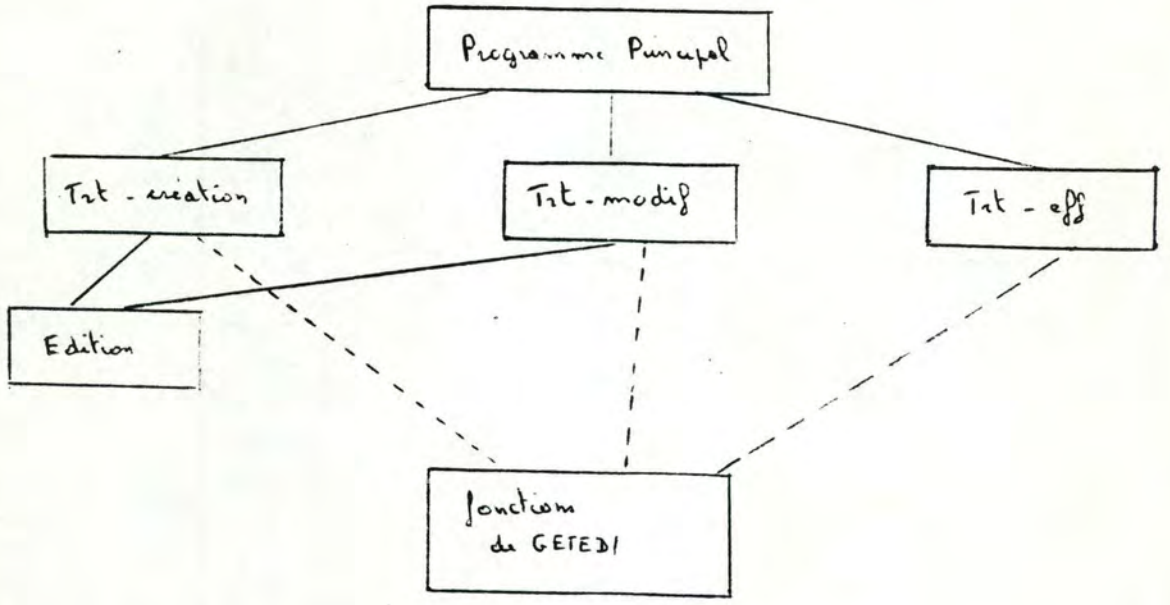
La sémantique des commandes est la suivante :

CREATION : L'utilisateur doit introduire les dimensions de l'écran à créer. Ensuite, il peut effectuer les opérations d'édition. A la fin, il a la possibilité de sauver la fenêtre créée sur fichier.

MODIFICATION : L'utilisateur doit introduire le nom de la fenêtre à modifier et le nom du fichier sur lequel elle se trouve. Ensuite, il peut effectuer les opérations d'édition. A la fin, il a la possibilité de sauver les opérations effectuées.

EFFACAGE : L'utilisateur doit introduire les noms des fenêtre et fichier où il veut effectuer la suppression.

2.2.4.2. Structure des modules



Les trois modules trt_création, trt_modif et trt_ ff traitent l'ensemble des opérations nécessaires pour la création, modification et l'effaçage de fenêtres (lecture des paramètres, mise en oeuvre des routines nécessaires)

Procédure edition

- IN : curwin : désignateur d'une fenêtre

  commandes au terminal

- OUT : curwin modifiée

- Sémantique : effectue sur curwin les opérations demandées par l'utilisateur au terminal :
  C-N, C-P, C-F, C-B : mouvements de curseur vers le bas, le haut, en avant ou en arrière.
  C-A, C-E, C-U, C-D : mouvements de curseur en début de ligne, en fin de ligne, en haut de la fenêtre ou au bas de celle-ci.
  C-X suivi de n,i,s,g,c : commence l'introduction d'un champ respectivement en mode d'affichage vidéo normal, inverse, souligné, gras ou clignotant. L'introduction est terminée par RETURN
  w : efface l'écran, sans toucher au contenu de la fenêtre
  e : efface le contenu de la fenêtre, sans toucher à l'écran
  c : efface l'écran et le contenu de la fenêtre
  C-L : provoque le réaffichage de l'écran tel que le connaît le gestionnaire
  t : marque toute la fenêtre comme modifiée
  r : provoque un refresh de la fenêtre courante
  B : trace un cadre autour de l'écran
  C-H suivi de 'c' ou d'un autre caractère : provoque un "hard-

copy" de l'écran sur imprimante, respectivement avec cadre ou sans.

2.2.5. Critères de structuration

La structuration en modules s'est effectuée en se basant sur trois critères :

1. **Un module doit avoir une forte cohésion interne.**

   Les modules tels qu'ils sont définis ont chacun une tâche bien définie : introduction de paramètres, contrôle de validité, opérations d'édition sur un écran...

2. **Un module doit avoir un faible degré de couplage vis-à-vis des autres modules.**

   On a essayé de simplifier au maximum les interfaces entre les modules : commandes mises sous forme interne, conditions de terminaison indiquées par un ensemble de codes retour bien défini,...

3. **Chaque module doit avoir un "secret", cacher une information** (façon d'implémenter quelque chose, algorithme employé,...)

   La façon dont se fait le "decodage" des strings en commande est cachée dans les procédures *_decode (p.ex., les contrôles effectués sur la date et la conversion de celle-ci sont cachés dans le module date_decode).

   La façon d'introduire les paramètres est cachée dans les modules intro_*.

2.3. Décisions d'implémentation

2.3.1. Choix du langage

Le choix du langage a d'abord été dicté par la contrainte "gestion d'écran". Il fallait disposer d'un gestionnaire compatible avec le langage choisi. De plus, ce gestionnaire devait pouvoir fonctionner avec tout le parc des terminaux de Namur(le gestionnaire TRAFFIC-20 de COBOL ne respectait pas cette deuxième contrainte). Nous disposions d'un tel gestionaire d'écran écrit en C. Malheureusement, une mauvaise adaptation du gestionnaire à TOPS-20 (le gestionnaire et le compilateur proviennent de UNIX), des problèmes avec le compilateur, s'ajoutant à une maîtrise imparfaite du langage, n'ont pas permis le développement dans cette direction.

Le choix s'est donc reporté sur PASCAL : c'est un langage que nous maîtrisions bien, "structuré" et assez répandu. Dans sa version standard, Pascal souffre de quelques grosses lacunes, e.a. dans le domaine de la gestion des fichiers et des "chaînes de caractères". Néanmoins, dans la version disponible à Namur (et dans la grande majorité des cas), ces "oublis" ont été corrigés, le seul problème étant la diversité de ces solutions.

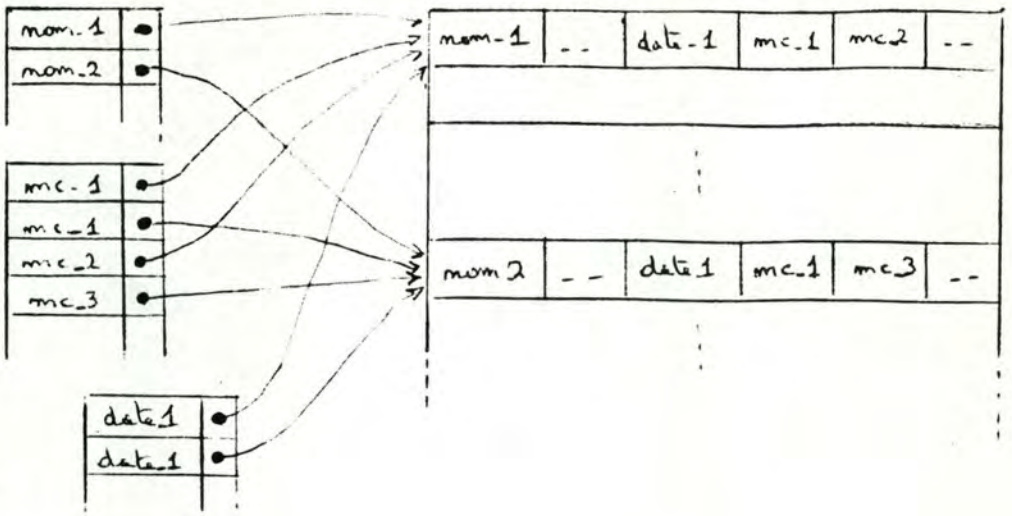
2.3.2. Décisions particulières

2.3.2.1. AIDE

1. **Représentation de la collection des fiches signalétiques.**

   On représente la collection des fiches signalétiques par un fichier dont l'article est un élément de la collection. L'ordre des articles est non significatif.

   Pour pouvoir accéder aux articles de fichier par nom de programme, date de mise-à-jour ou mot-clé, on crée trois fichiers auxiliaires d'index. Les articles de ce fichier sont constitués d'un nom, d'une date ou d'un mot-clé, accompagné de l'"adresse" dans le fichier principal de la fiche signalétique correspondante. Les articles de ces fichiers sont triés par valeur de "clé".



Pour l'accès par nom, on recherche séquentiellement le nom en question et on obtient par accès direct la fiche signalétique.

Pour l'accès par date, on recherche séquentiellement la première date postérieure à la date demandée, et on obtient les renseignements souhaités de toutes les fiches "pointées" par les éléments suivants.

Pour l'accès par mot-clé : soit on trouve un mot-clé égal à celui cherché, et on peut afficher les renseignements correspondants;
soit on trouve un mot-clé similaire : si on l'accepte, on peut obtenir les renseignements correpondants, sinon, on continue la recherche.

Afin d'accélerer les recherches, les fichiers d'index sont chargés en MC au début des opérations.

2. Représentation de l'index en MC

On convertit les fichiers d'index en listes linéaires, implémentées sous forme de chaîne de pointeurs (les tableaux, devant avoir des bornes fixes, ne présentent pas une souplesse suffisante).

Cette implémentation est cachée au sein de procédures qui forment une "interface abstraite" entre l'implémentation de la liste et le reste de l'application. Cette interface est suffisamment générale pour que l'implémentation de la liste puisse changer sans que le reste de l'application ne doive changer.

Les fonctions disponibles sur les listes sont la création, l'insertion d'élément, la suppression, une fonction fournissant le suivant d'un élément, des procédures permettant de manipuler le contenu d'un élément.

Pour implémenter cette interface, on serait tenté de créer un type "liste de X", sur lequel on définirait des opérateurs tels que création, suppression, etc. Malheureusement, un tel mécanisme de "type de donnée abstrait" n'est implémenté sur aucun langage courant, et a fortiori pas sur Pascal. Nous avons donc été réduit à créer un ensemble de fonctions et procédures pour chaque type de liste employé, ce qui est évidemment très lourd.

(Les types de liste utilisés sont :

- Liste de mots-clés, liste de dates, liste de noms : pour utiliser en mémoire centrale les index vers le fichier des fiches signalétiques.

- Liste de "résultats" : liste constituée comme résultat d'opérations de recherche sur les divers index; elle est constituée d'"adresses" de fiches signalétiques dans le fichier principal.)

3. **Mécanisme des codes-retour et des messages.**

Les procédures et fonctions composant AIDE signalent leur condition de terminaison en affectant une valeur à un code retour. Ces codes retour sont implémentés sous forme d'un type Pascal constitué de l'union de toutes les valeurs de code retour possibles.

Le programmeur doit prendre les mesures nécessaires quand le code retour indique une condition de terminaison spéciale; souvent, parmi ces actions figure l'affichage d'un message prévenant l'utilisateur de ce qui s'est passé. Afin de faciliter cela, nous avons mis sur pied le mécanisme suivant : à chaque code retour qui le nécessite, on peut associer un message dont le programmeur peut demander l'affichage. Ces messages sont stockés dans un fichier séparé et chargés dans une table où les valeurs des codes retour servent d'indice; ainsi, il est possible de modifier les messages (pour les traduire par exemple) avec beaucoup de souplesse.

Vu sa souplesse, le mécanisme que nous venons de décrire a été étendu à des codes ne provenant pas de procédures, mais créés pour pouvoir disposer des facilités offertes par ce système de messages.

Les codes retour peuvent se classer en quatre catégories :

- les codes d'erreur

- les codes de "message" ou de "question"

- les codes de demande de confirmation

- les codes sans message associé

Cette classification permettra de fournir des procédure de traitement de messages "intelligentes", càd tenant compte du type de message qu'elles traitent.

4. **Mode d'affichage**

On est soumis à la contrainte "gestion d'écran"

5. **Mode d'introduction des commandes** Afin d'offrir une interface utilisateur qui puisse aider le novice sans ralentir l'utilisateur chevronné, on a opté pour un système hybride menu/langage de commande.

Au départ, l'utilisateur se trouve devant un menu plein écran; en bas de l'écran, une ligne où il peut introduire son choix;

l'utilisateur habitué au système peut y introduire une commande complète, après quoi le système effectue la commande demandée. Le novice peut lui n'introduire que le mode d'accès souhaité; un deuxième écran apparaîtra, situé dans les trois premières lignes de l'écran du terminal, et l'invitera à compléter sa commande; après quoi, la réponse du système apparaîtra dans la partie basse de l'écran.

6. **Traitement des dates**

Les procédures d'introduction et d'analyse de date permettent l'introduction de la date sous toute forme respectant la contrainte JOUR-MOIS-ANNEE.

En format interne, les dates sont représentées sous forme AAAA/MM/JJ, ce qui permet un tri et une recherche chronologique.

7. **Aide**

A tout moment où l'utilisateur doit introduire quelque chose, il peut invoquer de l'aide en tapant "?"

2.3.2.2. INTRO

1. **Représentation de la collection des fiches signalétiques**

La décision prise au niveau d'AIDE d'implémenter la collection des fiches signalétiques par un fichier à accès direct et trois fichiers auxiliaires d'index a des implications au niveau d'INTRO, qui devra gérer cet ensemble de fichiers.

Pour faciliter la manipulation des index, les fichiers auxiliaires sont transférés en MC, sous forme de listes linéaires implémentées par chaîne de pointeurs.

2. **Tri**

L'algorithme de tri des fichiers auxiliaires est isolé dans un module physique séparé (ce qui permet de le modifier facilement). Pour le tri, les éléments des fichiers sont chargés en MC sous forme d'une table de pointeurs vers les éléments. Cette table est triée par déplacement de pointeurs, selon un algorithme simple. Après le tri, le fichier est reconstitué. (Vu la rigidité du mécanisme de type de Pascal, nous avons du créer trois versions de la procédure de tri, ne différant entre elles que par le type de liste traitée)

3. **Entrées et sorties**

Pour ce programme "utilitaire", les entrées/sorties se font par des instructions de base Pascal (read et write), sans gestion d'écran.

2.3.2.3. GETEDI

1. **Représentation des fenêtres :**

Les fenêtres sont implémentées sous forme d'un record Pascal. On dispose aussi de pointeurs vers ces records, ce qui permet d'utiliser l'allocation dynamique de mémoire pour créer et détruire les fenêtres.

Ce record regroupe les informations suivantes :

- coordonnées minimales, maximales et courantes de la fenêtre
- booléen indiquant si la fenêtre a été marquée pour l'effacement
- tableau de booléens indiquant si chaque ligne a été modifiée ou pas
- nom de la fenêtre.

Il reste à représenter le contenu même de la fenêtre. Pour chaque position de l'écran, on doit connaître le contenu et le mode d'affichage vidéo de celui-ci. Deux solutions sont envisageables : représenter les "champs" de la fenêtre par une chaîne de "strings" reliés par des pointeurs, ou représenter l'image de l'écran par un grand tableau de caractères.

La première solution est économique en espace mémoire, mais conduit à des opérations très lourdes dès qu'on veut modifier une fenêtre, ou superposer deux écrans. Pour cette raison, la seconde solution a été retenue. Pascal ne connaissant pas les tableaux à bornes variables (sauf en tant que paramètres de procédures), cette décision implique que les dimensions maximales des écrans que peut traiter le gestionnaire doivent être fixées à la compilation.

La première idée qui vient à l'esprit pour représenter ainsi une fenêtre est d'employer un tableau à deux dimensions de "positions d'écran", chaque position ayant un contenu et un attribut d'affichage. Malheureusement, l'implémentation de Pascal que nous avons à notre disposition ne sait stocker sous forme compacte (plusieurs éléments par mot mémoire) que des tableaux de caractères. Aussi, chaque tableau de "positions", composé d'éléments plus complexes, occupait une place prohibitive. Cette difficulté a été contournée en utilisant deux tableaux "parallèles", le premier pour représenter les caractères, le deuxième pour stocker leur mode d'affichage.

Cette implémentation est cachée à l'utilisateur (et même à la plupart des routines du gestionnaire) par une 'interface abstraite' composée d'une série de procédures et fonctions permettant d'accéder aux différents éléments sans en connaître la représentation interne.

2. **Sauvetage des fenêtres :**

On utilise un fichier séquentiel, dont le format d'enregistrement est identique au format de représentation interne des fenêtres.

3. Gestion des terminaux au niveau physique

Cette gestion est isolée dans le module (physique) TSCR, dont on ne connaît que l'interface abstraite.

Cette interface est composée des fonctions suivantes :

```
PROCEDURE xiscr(VAR code_ret : typ_code_ret);
PROCEDURE xrscr;
                Procédures d'initialisation et de cloture de
                la gestion d'ecran.
PROCEDURE xclli;
PROCEDURE xcltoe;
PROCEDURE xcls;
PROCEDURE xclear(debx,deby,finx,finy : INTEGER);
                Procédures d'effacement, respectivement du
                curseur à la fin de la ligne, du curseur à la
                fin de l'écran, de tout l'écran, et enfin de
                l'écran entre les limites précisées.
PROCEDURE echo;
PROCEDURE noecho;
                Procédures établissant ou supprimant l'écho
                des caractères tapés au clavier sur l'écran.
PROCEDURE xso(rendition : typ_rendition);
PROCEDURE xse;   Procédures établissant ou supprimant les
                caractéristiques vidéo particulières des
                caractères affichés. "rendition" peut valoir
                "blink" (caractères clignotants), bold
                (caractères en surintensité vidéo), "reverse"
                (caractères en vidéo inverse) ou "under"
                (caractères soulignés).
FUNCTION xphysx : INTEGER;
FUNCTION xphysy : INTEGER;
                Fonctions fournissant les dimensions du ter-
                minal physique courant, en abscisse et en
                ordonnée.
PROCEDURE xmove(x,y : INTEGER);
PROCEDURE xcu;
PROCEDURE xcd;
PROCEDURE xcf;
PROCEDURE xcb;   Procédures de mouvement absolu ou relatif du
                curseur, vers le haut, le bas, en avant ou en
                arrière.
```

4. Code retour et messages

Les messages sont implémentés par le système décrit ci-dessus pour AIDE.

2.4. Implémentation

2.4.1. Principes généraux

Les critères de structuration utilisés pour la décomposition générale en modules ont été aussi utilisés, dans la mesure du possible, pour l'implémentation en "procédures" et "fonctions". (Ainsi, le principe des 'interfaces abstraites' que nous avons évoqué plus haut, e.a. au niveau du traitement des listes, est une application du critère qui parle de "cacher de l'information".) Cette façon de décomposer le travail facilite la compréhension et la maintenance du logiciel, favorise la portabilité (en isolant les parties non-portables) et permet un travail de développement par étapes.

Exemples :

Gestionnaire d'ecran : les parties liées à un matériel spécifique étant cachées à un niveau relativement bas, le gestionnaire a d'abord été développé avec des routines physiques provisoires, convenant juste pour le terminal sur lequel s'est effectué le développement. Une fois le gestionnaire terminé et testé, l'adaptation aux autres terminaux s'est faite sans grandes difficultés.

Programme d'aide : le programme a d'abord été développé sans gestion d'écran, avec une interface utilisateur provisoire. Après que les tests fonctionnels aient été réalisés, on a ajouté la gestion d'écran.

La chronologie du développement a été la suivante :

- développement de la version mono-terminal, puis de la version universelle du gestionnaire d'écran.
- développement de différentes versions de plus en plus complètes de l'éditeur d'écran SCREEN.
- développement de CONVDA, routines de traitement de dates, et de DACCES, routine permettant, dans un cadre de gestion d'écran, d'afficher le contenu d'un fichier de façon "dynamique" (c-à-d qu'il est possible d'avancer et de reculer dans le fichier). Ces routines ont été testées en stand-alone à l'aide de petits programmes auxiliaires.
- développement de INTRO et de AIDE sans gestion d'écran.
- ajout de la gestion d'écran à AIDE.

2.4.2. Implémentation des "utilitaires"

Afin de faciliter la programmation des logiciels à proprement parler, et de pallier à certaines lacunes de Pascal, nous avons créé quelques "utilitaires".

2.4.2.1. Utilitaires divers

D'abord, une série de petites routines diverses, essentiellement dans le domaine de la gestion de chaînes de caractères :

Fonction confirm;
vaut "true" si le caractère donné en argument est "o" ou "O" ou "y" ou "Y", "false" sinon.

Procédure bell; actionne le beeper du terminal.

Procédure getc;
saisit un caractère au vol au clavier (sans attendre le RETURN)

Fonction n_to_s;
convertit le chiffre "i" (0< = i < = 9) en le caractère ASCII le représentant.

Fonction s_to_n;
convertit un string constitué de caractères numériques en le nombre équivalent.

Fonction erstat;
retourne le numéro d'erreur lors de l'ouverture d'un fichier (0 si pas d'erreur).

Fonction alpha; vérifie qu'un string est constitué de caractères alphabétiques.

Fonction num; vérifie qu'un string est constitué de caractères numériques.

Fonction match; vérifie si deux strings correspondent sur leur longueur.

Procédure wstr; écrit un string sur un fichier sans les blancs de fin.

Procédure discard;
lit un fichier tant qu'il s'agit de caractères d'un ensemble déterminé.

Procédure lower;
convertit un string en minuscules.

Procédure concat;
concatène deux strings.

Fonction simili;
teste la similarité de deux chaînes de caractères .

Cette dernière fonction est la seule un peu particulière dans cet ensemble d'utilitaires; en voici l'algorithme

```
i:=0 ; j:=0 ;
diff := 0;
WHILE (i < n_car_max) AND (j < n_car_max) AND (diff >= 0) DO
      i := i + 1;
      j := j + 1;
      IF str1[i] ≠ str2[j] THEN
            diff := diff + 1;
            i := i + 1;
            IF str1[i] ≠ str2[j] THEN
                  i := i - 1;
                  j := j + 1;
                  IF str1[i] ≠ str2[j] THEN
                     i := i + 1;
                     IF str1[i] ≠ str2[j] THEN diff := -1;
                  FI;
            FI;
      FI;
OD;
```

Cet algorithme compte les "écarts" entre deux chaînes de caractères; on considère qu'il y a un "écart" quand les deux chaînes de caractères diffèrent par un caractère mais que le caractère qui suit est le même.

2.4.2.2. Utilitaire de traitement de dates

Ces procédures sont groupées dans un module physique, ce qui permet le cas échéant de les modifier facilement.

Nous disposons de deux procédures :

- Conversion d'une date introduite par l'utilisateur en date interne (la date peut être introduite sous toute forme telle que "jour" est avant "mois" est avant "année")
- Conversion d'une date interne en une date au format agréable pour l'utilisateur.

La première de ces deux procédures est la plus complexe; elle dispose de deux routines auxiliaires : la première qui permet de vérifier qu'une année est bissextile et la seconde qui convertit un mois "en toutes lettres" en un mois sous forme numérique.

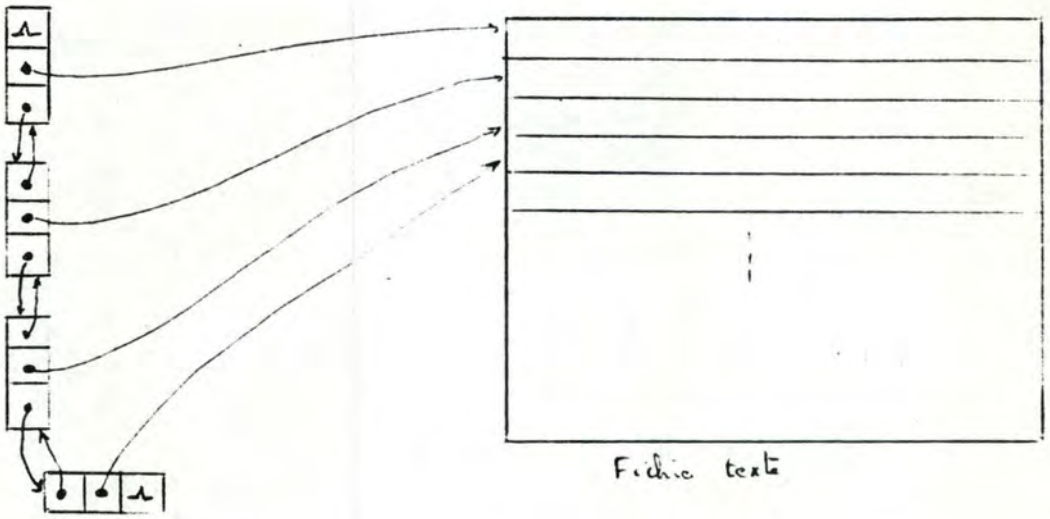
Cette procédure est composée de deux parties :

- analyse syntaxique (le jour doit être un nombre de deux chiffres maximum, le mois sous forme numérique de même, le mois en toutes lettres a dix caractères au maximum, l'année doit être un nombre de quatre chiffres)
- analyse sémantique (le nombre de jour doit être correct par rapport au mois et à l'année, la date doit être antérieure à la date du jour)

2.4.2.3. Utilitaire d'affichage de fichier en gestion d'écran

Cet utilitaire permet d'afficher à l'écran un fichier, l'exploration pouvant se faire en avant ou en arrière.

La routine est basée sur un ensemble de "positions" (= pointeurs vers un endroit dans le fichier accédé en accès direct), reliées par une double chaîne de pointeurs.



La chaîne de pointeurs est constituée au fur et à mesure que l'utilisateur avance dans le fichier. L'intervalle entre deux positions représente le contenu d'un écran. Les positions sont prises de telle façon qu'il y ait un recouvrement de N lignes (dans le cas présent, N = 2) entre deux écrans. La chaîne, une fois constituée, permet de retrouver les positions précédentes et donc d'explorer le fichier en "marche arrière".

2.4.3. Gestion de listes linéaires

Les fonctions et procédures qui composent l'interface abstraite sont toutes assez simples; en voici la liste :

fonction X_cree : crée une liste de X

fonction X_suivant(liste,element) : si "element" est vide, fournit le premier de la liste de X "liste"; sinon, fournit l'élément suivant "element" dans "liste".

procedure X_content(element) : si "element" n'est pas vide, fournit la valeur de la partie 'content' du contenu de celui-ci; sinon, fournit un string de valeur ASCII maximale.

procedure X_adresse(element) : si "element" n'est pas vide, fournit la valeur de la partie 'adresse ' du contenu de celui-ci; sinon, fournit la valeur MAXINT

procedure set_X_content (element, contenu) : assigne la valeur "contenu" à la partie 'content' de l'élément "element".

procedure X_ins(liste, prec, contenu) : insère un élément de contenu "contenu" dans la liste de 'X', désignée par "liste", après l'élément désigné par "prec" si "prec" n'est pas vide, en debut de liste si "prec" est vide.

procédure X_discard(liste) : détruit la liste dont le désignateur est donné en argument. Après cette procédure, la liste vaut VIDE.

procédure X_supp(liste, element) : détruit l'element "element" de la liste "liste" .

Ces procédures étant définies avec X valant lmc,ldem,lres,ldate,lnom.

Voici l'algorithme de trois d'entre elles :

- Insertion (après un élément donné)

```
créer un nouvel élément
nouvel_él := élément à insérer
SI él_prec = VIDE
   ALORS suivant(nouvel_él) := tête de liste
         tête de liste := nouvel_él
   SINON suivant(nouvel_él) := suivant(él_prec)
         suivant(él_prec) := nouvel_él
```

- Suppression de liste

```
courant := tête de liste
TANT QUE courant ≠ VIDE FAIRE
      él_suivant := suivant(courant)
      supprimer(courant)
      courant := suivant(courant)
liste := VIDE
```

- Suppression d'un élément de liste.

```
SI él à supprimer = tête de liste
 ALORS tête de liste := suivant(tête de liste)
      supprimer(él à supprimer)

SINON courant := tête de liste
  TANT QUE courant ≠ él à supprimer
                   ET courant ≠ VIDE FAIRE
     précédent := courant
     courant := suivant(courant)
     SI courant = VIDE
       ALORS épas trouvé l'élémentè
       SINON suivant(précédent) := suivant(courant)
             supprimer(él à supprimer)
```

Toutes ces procédures ne sont pas forcément implémentées pour tous les programmes utilisant des listes; par exemple, AIDE, ne faisant que de la consultation, n'a pas besoin des fonctions de suppression de liste de dates, de noms ou de mots-clés.

2.4.4. AIDE

Dans l'ensemble, les algorithmes sont relativement "triviaux"

- recherche dans une liste
- affichage
- introduction de commandes
- décodage de chaînes de caractères en date, nom de programme ou liste de mots-clés

Le seul algorithme un peu délicat est celui qui recherche dans une liste d'index par mots-clés les éléments correspondants à un ou des mots-clés cherchés (appelés "demandes") et produit une liste de ces éléments (appelée "liste de résultats").

Voici cet algorithme :

```
dem_cour := première demande
TANT QU'il y a des demandes FAIRE
  anc_liste_rés := liste_rés
  endroit_début_recherche := tête de la liste des mots-clés

  REPETER

    recherche d'un mot-clé similaire ou égal
            (à partir d'endroit_début_recherche)

    SI on a trouvé un mot-clé égal
       ALORS endroit_début_recherche := suivant(mc_trouvé)
                (* de là, on continuera la recherche pour trouver
                   les autres mots-clés  *)
             insertion := TRUE
                (* il y aura un mot-clé à insérer *)

    SI on a trouvé un mot-clé similaire
       ALORS on demande confirmation
             SI confirmé (* on considère la demande courante comme
                            un mot-clé déjà existant, mal
                            orthographié : il ne faut pas insérer
                            2 versions différentes d'un même
                            mot-clé et on "corrige" donc la demande
                            courante *)
                 ALORS dem_cour := contenu(mc_trouvé)
                       insertion := TRUE
                       endroit_début_recherche :=
                                    suivant(mc_trouvé)
             SINON
                endroit_début_recherche est mis après tous les
                  mots-clés égaux à celui que l'on vient de refuser

    SI insertion ALORS
       SI il n'y a ps d'ancienne liste de résultats
             ALORS on insère le mot-clé trouvé parmi les résultats
       SI il existe une ancienne liste de résultats
             ALORS on n'insère que si le mot-clé trouvé y est aussi

  JUSQU'A ce qu'on ne trouve plus de mots-clés similaires
  On élimine l'ancienne liste de résultats
  On prend la demande suivante
```

2.4.5. INTRO

L'algorithme d'INTRO ne présente aucune difficulté particulière;

Le principe de fonctionnement est le suivant :

En début d'exécution, les index par nom et mot-clé sont chargés en mémoire centrale; lors de l'introduction d'éléments, ces listes sont utilisées pour vérifier l'existence de doubles; lors des mises-à-jour, elles servent aussi à retrouver la fiche à modifier dans le fichier des fiches signalétiques.

En ce qui concerne le fichier principal, les ajouts se font par allongement du fichier à la fin (append); les mises-à-jour se font par lecture (get) puis réécriture au même endroit (put); (c'est possible car le fichier a été ouvert par "update"); les suppressions se font en mettant une marque sur la fiche supprimée.

En ce qui concerne les index associés, les créations, modifications et suppression se font en MC.

En fin de traitement, le fichier principal est recopié en omettant les éléments marqués pour la suppression, et, simultanément, les fichiers auxiliaires sont reconstitués et ensuite triés.

N.B. : Par rapport à la structure fonctionnelle vue plus haut, il faut ajouter un module supplémentaire chargé de trier les fichiers auxiliaires d'index.

2.4.6. GETEDI

Le gestionnaire pourrait être décomposé en deux ou trois couches :

- La première couche est la couche physique, utilisant TSCR.
- La deuxième couche s'appuie sur cette couche de base pour assurer la gestion au niveau d'écrans logiques : création, effacement, ajout de caractères, etc.
- Une troisième couche pourrait être une couche où le gestionnaire traiterait des "objets spéciaux" qui seraient des écrans avec un contenu ("sémantique") spécialisé, avec des tritements particulier : grilles de saisie, menus,... Dans l'état actuel des choses, cette couche est embryonnaire et ne fournit que quelques routines de traitement de "messages".

Le principe de base classique sur lequel repose un gestionnaire d'écran est l'existence d'une "référence" qui représente la vue que le gestionnaire a de l'écran. Pour GETEDI, il s'agit d'une fenêtre "curscr", qui a la taille de l'écran du terminal physique sur lequel tourne l'application. Cette fenêtre permet au gestionnaire de ne rafraîchir que les parties de l'écran qui le nécessitent, ainsi que de restaurer un écran perturbé par un message en provenance du système.

Voyons quelques points particuliers :

1. **Algorithme de rafraîchissement d'une fenêtre**

```
SI la fenêtre est marquée pour l'effacement
OU SI on rafraîchit curscr (on dira "refresh forcé")
     ALORS on efface l'écran sur l'étendue de la fenêtre
SI la fenêtre est marquée pour l'effacement
     ALORS on met à blanc Curscr

Pour toute ligne de la fenêtre

  SI elle est marquée comme modifiée
  OU SI c'est un refresh forcé ALORS

     Pour tout caractère dans la ligne

      SI c'est un refresh forcé ET  caractère ≠ ' '
      OU SI caractère ≠ caractère dans Curscr

          ALORS s'il n'y est pas encore, on place le
                    curseur à la bonne position
               on met en oeuvre le mode d'affichage
                    vidéo ad hoc
               on réécrit le caractère
               on met à jour Curscr

    On marque la ligne comme non modifiée

On place le curseur à la position courante telle qu'elle
     est définie dans la fenêtre.
```

2. **Initialisation**

La procédure d'initialisation effectue les opérations suivantes :

- Chargement des messages correspondants aux codes-retour dans la table des messages
- Initialisation de diverses variables globales du gestionnaire
- Initialisation de la gestion d'écran physique
- Définition des lignes de message
- Création de Curscr

3. **Procédure d'entrée à partir du terminal**

L'entrée de caractères à partir du terminal nécessite certaines précautions, quand on travaille en gestion d'écran. Il faut veiller à trois problèmes :

a. Si, dans un écran de N colonnes, on introduit N+1 caractères, certains terminaux effectuent un retour à la ligne

automatique, ce qui détruit l'ordonnancement de la suite de l'écran. Cette caractéristique, ("auto linefeed") est contrôlable par software sur la majorité des terminaux modernes; malheureusement sur des terminaux plus anciens, cela n'est pas possible. Il faut donc limiter le nombre de caractères qu'il est possible d'introduire à N.

b. La façon classique de faire un "delete" est de reculer le curseur d'une position (backspace), puis de faire un "kill to end-of-line"; cette façon de faire pose des problèmes quand, après le champ sur lequel on travaille, il y a un deuxième champ sur la même ligne. Pour cette raison, "delete" a été redéfini comme "backspace, écriture d'un blanc, backspace".

c. Les caractères de tabulation posent un problème, car ils rompent la synchronisation "nombre de caractères - nombre de positions sur l'écran", ce qui est gênant en gestion d'écran. On remplace donc les caractères de tabulation par un nombre de blancs équivalent.

4. **Procédures d'ajout de caractères à l'écran**

Pour toutes les fonctions ajoutant des caractères à l'écran, il faut veiller aux problèmes posés par le nombre de caractères introduit et les caractères de tabulation.

2.5. Extensions de Pascal utilisées

Dans la toute grande majorité des cas, dès que l'on commence à développer un logiciel un tant soit peu conséquent, le Pascal standard se révele "inapte au service"; on est donc conduit à utiliser des extensions, ajoutées par quasi tous les concepteurs de compilateur.

Dans le cas présent, les principales extensions de Pascal qui ont été employées et qui limitent donc la portabilité sont :

- Appels systèmes pour la gestion des terminaux physiques
- Tableaux à bornes dynamiques comme paramètres de procédure; (cette possibilité n'existe pas dans le Pascal standard de Jensen et Wirth, mais fait partie du standard Pascal de l'ISO.)
- Extensions dans le domaine de la gestion des fichiers :

  * Ouverture de fichiers avec gestion des erreurs par le programmeur (ceci permet, par exemple, de tester l'existence d'un fichier)
  * Accès direct sur base d'un numéro de record
  * Ouverture d'un fichier en ajout (append)
  * Changement de nom, effaçage d'un fichier (rename et delete)

* Entrées-sorties au terminal

  - 'READ' modifié de façon à ce qu'on dispose du nombre de caractères lus et que le programmeur puisse déterminer l'ensemble des caractères provoquant la fin de la lecture

  - Opérations sur des 'strings' (tableaux de caractères) similaires aux opérations sur fichier

- Mécanisme des 'INCLUDE'. Ce mécanisme permet d'inclure en tête d'un programme source Pascal un ou des fichier(s) contenant des déclarations de constantes, de types ou de procédures externes.

  Ce mécanisme est utilisé de deux façons :

  * Déclaration de types et de constantes utilisées par plusieurs programmes :

    - elems.dec : déclaration des éléments des listes linéaires utilisées par AIDE et INTRO

    - string.dec : déclaration d'une série de "strings" str** (tableaux de caractères) de longueur fixe, des constantes bl** (à blanc) et maxstr** (valeur ASCII maximale) correspondantes

    - cod_ret.dec : regroupe toutes les déclarations des "code-retour" des différents programmes

  * déclarations des types et procédures des librairies communes :

    - utilit.dec : utilitaires, essentiellement dans le domaine de la gestion des chaînes de caractères

    - getedi.dec : gestionnaire d'écran

    - tscr.dec : gestion d'écran au niveau physique

Ce sont les extensions au niveau des fichiers qui poseront le plus de problèmes du point de vue de la portabilité. Ces nouvelles possibilités se retrouvent sur la majorité des autres compilateurs, mais sous une autre forme. L'idéal aurait été de cacher ces différences dans un module 'Gestion de fichiers', mais le mécanisme de "types" de Pascal n'est pas assez souple pour cela. (Il ne connaît pas le type générique 'fichier')

2.6. Critiques et améliorations possibles

Tout en restant dans la ligne du travail accompli et sans remettre en cause les spécifications de base, un certain nombre de critiques peuvent être émises et des améliorations correspondantes proposées.

Ces critiques et améliorations relèvent de deux domaines :

1. Développements du logiciel tel qu'il est implémenté en Pascal
2. Développements qui seraient rendus possibles par un hypothétique langage X "idéal"

Continuation du développement du logiciel en Pascal :

- La liaison entre les procédures s'occupant de l'acquisition de paramètres en gestion d'écran et les fenêtres proprement dites est beaucoup trop forte. Dans l'état actuel des choses, ces routines doivent connaître l'adresse physique de l'endroit de l'écran où elles doivent faire la lecture. Si la fenêtre change, la procédure doit changer aussi.
  Il faudrait donc développer la "troisième couche" du gestionnaire d'écran, afin de lui permettre de manipuler des "grilles d'écran" (l'application ne connaîtrait que le nom de la fenêtre et les paramètres qui doivent y être lus; le reste serait isolé dans les routines du gestionnaire).
- Les interfaces homme/machine d'INTRO et de SCREEN pourraient être développées (par exemple, introduction de la gestion d'écran pour l'introduction et la mise à jour des fiches signalétiques).
- La gestion des fichiers est assez archaïque et de bas niveau; il serait probablement possible de créer des routines de plus haut niveau, malgré les limitations de Pascal. (Bien qu'un séquentiel indexé classique aurait rendu difficile la recherche par mot-clé "similaire")
- L'algorithme de comparaison de mots-clés par similarité est assez simple, si pas simpliste; cela présente l'avantage d'être efficace, ce qui est nécessaire vu que cet algorithme est utilisé intensivement; mais cela implique aussi que l'algorithme risque de considérer comme "similaires" deux mots qui, pour un utilisateur "humain", ne le sont pas du tout (pour prendre un exemple caricatural et imaginaire, "tableau" et "karlian" sont considérés comme similaires). En pratique cependant, ce genre de cas extrême arrive rarement; néanmoins, si on veut être perfectionniste, il existe des algorithmes plus évolués pour ce genre de comparaison.
- L'éditeur SCREEN pourrait être développé (ajout de fonctions "à la Emacs").

Si maintenant, on se met à rêver d'un langage "idéal", beaucoup de possibilités s'ouvrent :

- On pourrait implémenter les fenêtres et les listes par des "types de donnée abstrait";
- Des macros remplaceraient avantageusement certaines procédures de bas niveau (exemple type : les procédures de l'interface abstraite vers les fenêtres) .
- Un mécanisme d'assemblage et d'"include" conditionnel faciliterait la tâche du programmeur qui veut utiliser la gestion d'écran.

2.7. Conclusions

Le travail réalisé a dépassé en ampleur ce qui avait été prévu, e.a. par l'obigation dans laquelle nous nous sommes trouvé de réécrire le gestionnaire d'écran; mais ce qui importe, c'est la "partie visible de l'iceberg", le logiciel AIDE, qui est destiné à être utilisé par tous.

Ce logiciel fournit les fonctions prévues (information par rapport à l'existant : liste des programmes par nom, accès par mot-clé; fourniture d'une liste des programmes changés depuis une date donnée). L'interface utilisateur présente les caractéristiques suivantes :

- Interface combinant menu et langage de commande
- Possibilité d'avoir de l'aide à tout moment
- Possibilité d'avoir une trace écrite de ce qu'on voit à tout moment
- Messages explicites, dans la langue de l'utilisateur, de format bien défini et constant
- Format d'introduction libre

La mise en oeuvre du système ne demande que la constitution des fiches signalétiques, travail réalisable même avec des moyens réduits.

L'utilisation des techniques de découpage en modules isolés les uns des autres, des interfaces abstraites, etc, favorisent la portabilité du logiciel.

Nous pensons donc avoir atteint le but fixé à ce travail, et espérons que l'expérience ne nous démentira pas.

Nous pensons que des recherches ultérieures devraient aller dans le sens d'une standardisation des formats de la documentation, et surtout d'un accès

sur base sémantique (à la limite, on touche alors le domaine de l'intelligence artificielle...); néanmoins, nous ne sommes pas sûr que de telles recherches se justifient (et soient possibles) dans le cadre limité d'une université telle que Namur.

REFERENCES

[1] Raymond C. HOUGHTON, jr.
Online Help Systems : A Conspectus.
Communications of the ACM 27(2):126-133, feb, 1984.

[2] A. F. DAWSON, M. J. COOMBS, and J.L. ALTY .
How to Improve Computer Advisory Services.
Software-Practice and Experience 12:857-877, 1982.

[3] J. L. ALTY, and M. J. COOMBS.
University Computing Advisory Services : The Study of the Man-Computer Interface.
Software-Practice and Experience 10:919-934, 1980.

[4] T. R. GIRILL and Clement H. LUKE.
Document : An Interactive, Online Solution to Four Documentation Problems.
Communications of the ACM 25(5):328-337, may, 1983.

[5] Laura L. SCHARER.
User Training : Less is More.
Datamation :175-176,181-182, jul, 1983.

[6] Jacob PALME.
A Human-Computer Interface for Non-computer Specialists.
Software-Practice and Experience 9:741-747, 1979.

[7] Michael J. HEFFLER.
Description of a Menu Creation and Interpretation System.
Software-Practice and Experience 12:269-281, 1982.

[8] D. SCHOFIELD, A. L. HILLMAN, and J. L. RODGERS.
MM/1, A Man-Machine Interface.
Software-Practice and Experience 10:751-763, 1980.

[9] James SNEERINGER.
User-interface Design for Text Editing : A Case Study.
Software-Practice and Experience 8:543-557, 1978.

[10] Saul I. GASS, Karla L. HOFFMAN, Richard H.F. JACKSON, Lambert S. JOEL, and Patsy B. SAUNDERS.
Documentation for a Model : A Hierarchical Approach.
Communications of the ACM 24(11):728-733, nov, 1981.

[11] Melinda THEDENS.
Cataloging the Program Library.
Datamation :247-250, may, 1983.

APPENDICE

I. RAPPORT DES INTERVIEWS

Une secrétaire dans le domaine de l'informatique

Travail: utilisation de logiciels d'intérêt général: Scribe, Emacs, MM

- L'apprentissage s'est fait par contact personnel avec des personnes compétentes et personnellement, par la consultation des manuels. Pour ces derniers, deux remarques:

  * La langue anglaise peut être un obstacle pour certaines personnes.

  * Il est nécessaire de présenter des exemples

- Dans l'apprentissage, les possibilités de la touche ' ? ' sont très pratiques.

Administration

Travail: administratif...

Utilisation de programmes réalisés "sur mesure" par le centre de calcul, donc pas de problème particulier de documentation.

Chercheur "mathématicien"

Travail: responsable de l'utilisation de Sophie pour les matheux (applications habituelles: beaucoup d'utilisation de librairies et de programmes provenant tant de l'extérieur que de la "maison").

- Le "public" de l'informatique en math est très varié: il comprend tant des personnes passionnées que des utilisateurs "contraints et forcés".

- La documentation actuelle n'est pas satisfaisante: elle peut suffire pour quelqu'un qui est habitué à l'usage du soft, mais pas pour le commun des mortels.

- Actuellement, il y a trois sources de documentation:

  * les livres et manuels(éventuellement à la bibliothèque)

  * la machine, sur les directories HLP: et DOC: ; ces directories nécessitent une classification; par exemple:

    - HELP serait une introduction de quelques pages, une sorte de vade-mecum, qui inclurait une référence vers une documentation plus complète et l'indication de personnes,

habituées au logiciel, qui accepteraient de renseigner les utilisateurs moins qualifiés.

- DOC contiendrait une documentation plus complète.

* la machine, sur des directories plus spécialisées (p. ex [mat.library])

- Xinfo est peu utilisé: il donne l'impression d'un labyrinthe(sic). On n'a pas l'habitude de la gestion d'écran. Deux autres inconvénients sont la nécessité d'un certain apprentissage, et l'absence de document final écrit. En fait, Xinfo est trop bien pour le commun des mortels...

- Il y a un problème de communication et d'harmonisation au niveau des programmes qui se créent ou qui existent déja, au niveau de ce qui se fait à divers endroits avec PCL, etc.

- La traduction de MM en FF est une expérience pas tout à fait réussie... on a tellement l'habitude des mots-clés en anglais !

- BBOARD: il y a le problème des messages qui ne nous concernent pas, et qui reviennent tout le temps car ils ne sont jamais lus... Il manque une commande "passé et consideré comme lu"

Doctorant en Sciences économiques.

Travail: Recherche et doctorat

- Expérience à Stanford: Il y existe trois centres de calcul: deux équipés d'IBM's, payants(des vrais dollars sonnants et trébuchants); un troisième, dont on va parler, équipé d'un DEC, et accessible aux étudiants qui ont un travail à effectuer.

  A l'inscription, chacun reçoit une farde de documentation, comprenant quelques feuilles sur les facilités de calcul et la procédure à suivre.

  La formation des utilisateurs est placée sous la responsabilité du directeur pédagogique du centre. Le password n'est accordé qu'après quatre heures de cours (audio-visuel) présentant les bases de l'exec, de l'éditeur, etc. A la fin du cours, chacun reçoit une documentation très complète: ainsi le recours ultérieur à une aide du centre de calcul est rare.

  Accès au système: Il y a 120 terminaux, répartis en 3 pools, distants de 5 minutes l'un de l'autre. Dans chaque pool, un terminal est affecté à la gestion d'une "file d'attente". Pour obtenir un terminal, on s'adresse à la file; s'il y a un terminal de libre, il est accordé au demandeur; sinon, le demandeur est placé dans la file, et le temps probable d'attente est affiché. Le budget de chacun est exprimé en "connect time" (le genre de travail effectué sur le DEC étant en général peu gourmand en temps CPU). Outre le budget global, il y a un quota de connect time par jour, quota qui varie selon la charge.

- La situation à Namur est très différente: l'accès aux terminaux est

régi par une file tout ce qu'il y a de plus informelle (quand ce n'est pas la loi du plus fort...). Quant à la documentation, elle est éparse et dépend fort du bon vouloir de celui qui assure l'initiation, ce qui a comme conséquences que l'on est conduit à s'adresser à des spécialistes du centre ou d'ailleurs pour des broutilles.

- Il y a un problème de documentation: quand quelqu'un veut faire des graphiques simples pour illustrer un mémoire ou une thèse, on lui répond: il n'y a qu'à lire DI-3000... qui est plutôt indigeste.

- On propose le système suivant: un "gourou" donnerait un cours d'introduction de 1 ou 2 heures. A l'issue de ce cours, on fournirait une documentation plus ou moins complète mais accessible (un livre, même de base, accroche moins s'il n'y a pas eu une introduction préalable)

- Bboard et MM sont bons pour les initiés, mais trop compliqués pour les autres.

- Xinfo: incompréhensible...

- Help on line: il y a un "fatras incompréhensible" entre DOC: , HLP: , ... De plus on ne sait pas ce qui existe dans DOC: ...

Assistants et chercheurs en biologie quantitative.

Travail: recherche, donc utilisation de logiciels spécialisés et autres (Emacs, Scribe, SPSS, BMDP, ...) et programmation en divers langages: Basic, Fortran, APL, ...

encadrement de "naïfs": chercheurs, mémorants, doctorants, étudiants, ayant besoin sporadiquement du DEC.

1. Utilisation de Sophie pour la recherche:

- Les deux moyens de documentation sont les manuels et le coup de téléphone au CC (le deuxième moyen étant plus rapide et plus efficace en cas de problème). Pour les manuels, la difficulté principale réside dans les mises-à-jour: on manque souvent de documentation(On se rend compte qu'une modification a eu lieu quand un programme qui a toujours tourné "se plante"...)

- Bboard a le mérite d'exister, mais n'est pas utilisé outre mesure

- MM n'est pas utilisé (on emploie MAIL pour envoyer le courrier !)

- Xinfo "a l'air bien", mais il faut lui consacrer un certain temps d'apprentissage.

- Emacs est fort utilisé actuellement. Néanmoins, lors de l'introduction d'Emacs sur le système, il a fallu un certain temps avant qu'il ne soit utilisé effectivement.

- Il y a un problème au niveau de la documentation: la documentation, telle qu'elle existe, est souvent sous forme de dictionnaire: l'essentiel est caché dans 300 pages d'explications détaillées... Il est difficile de trouver les deux jours nécessaires pour s'y mettre.

- Améliorations envisageables:

  * Prévoir un monitoring d'introduction par le centre de calcul ou quelqu'un d'autre, quand on installe un nouveau software.

  * Etablir, pour les logiciels déjà installés, une liste d'utilisateurs, de "personnes-ressource", qui accepteraient de répondre aux questions et de partager leur savoir-faire et leurs "trucs" avec de nouveaux utilisateurs (whouse, cfwhois)

  * Etablir une liste des logiciels existants.

2. Encadrement de "naïfs"

- Il y a un besoin urgent et vital d'une documentation de base pour différents logiciels, l'exec et les terminaux. Il faut quelque chose de facile à consulter, présentant l'essentiel, et donc rapide à étudier(Teach-emacs est bien fait, mais déjà trop long et complexe pour un débutant). Il faut aussi que cette documentation soit disponible sur papier. (Un papier est plus facile à consulter qu'un écran; et puis, les terminaux ne sont pas disponibles pour l'initiation, pendant la journée). Il faut fournir une information de base, présentant des procédures faciles, "à l'épreuve des c...".

- Propositions diverses visant à améliorer l'utilisation du Dec par les débutants (et les autres):

  * Prévoir un "autologout" après X minutes d'idle (pour les gens qui croient sortir du système en éteignant le terminal)

  * Fournir une liste des opérations couteuses en CPU

  * A chaque logout, imprimer une "facture" plutôt que simplement le temps CPU et connect utilisé

  * Prévoir une information sur les terminaux (papier à coller sur la table de travail !)

Chef de travaux en médecine.

Travail: divers programmes d'intérêt plus ou moins local. Les programmes, informations, et tuyaux se transmettent de personne à personne.

- Les logiciels utilisés sont des logiciels d'intérêt général (Emacs,

Scribe). La documentation est fournie par les manuels, et, en cas de problème, par contact direct avec le CC.

- Indice de malaise du point-de-vue communication: il réalise une traduction de Teach-emacs en français... Personne ne le sait en dehors de son département... et puis, cette traduction existe déjà.

Dactylo novice en informatique.

Droit

Travail: introduction de données pour une base de données sur les compétences régionales et une autre sur la jurisprudence : utilisation d'Emacs sur DEC.

- Elle était totalement néophyte en informatique il y a 8 mois... L'initiation a été faite par contact avec une personne compétente.

- Sources de documentation: manuel Emacs, accompagné d'un bon dictionnaire anglais/français. A partir de son expérience et du manuel, elle s'est constitué un classeur de petites fiches cartonnées, reprenant les manipulations de base(y compris comment allumer le terminal), avec de petits dessins, des couleurs, des layouts d'écran, etc.

Assistant "orienté math" en informatique.

Travail: assistant "mathématicien" : un peu de traitement de texte, et des petits et moyens programmes (^{2}= 1000 l.).

Pas d'utilisation de logiciels, car il en a une très mauvaise expérience (peu précis, deux logiciels donnant des résultats différents sur les mêmes données...)

- Les logiciels existants sont tellement difficiles à utiliser (mal documentés) ou difficilement accessibles (sur bande) que c'est tout aussi rapide de réécrire le programme soi-même. De plus, les besoins diffèrent d'une fois à l'autre(plus ou moins de résultats plus ou moins détaillés)... alors on écrit du "throw-away code".

  C'est un cercle vicieux: on n'utilise pas des logiciels comme LAMPS car ils sont mal documentés, et, puisqu'on ne les utilise pas, on ne fait pas l'effort de mieux les documenter... (Il manque souvent un "How to get started with... ")

- Les machines ont fait des progrès énormes en puissance depuis 10 ans, mais du point de vue de l'utilisateur, on arrivait aux mêmes résultats avec des machines comme le Siemens ou le Bull... certains utilisateurs sont un peu désabusés...

Chercheur en informatique.

Travail: recherche sur le projet ISDOS : langage utilisé: Fortran. Pas besoin de documentation particulière(Les logiciels utilisés sont bien connus), sauf pour utiliser certaines caractéristiques 'exotiques'.

Un système genre "gazette" est intéressant pour savoir quand Sophie tourne, quand est-ce-que le temps CPU sera gratuit, etc...

En cas de problèmes particuliers, le plus rapide et le plus sûr est de s'adresser au CC. (Souvent le seul endroit ou l'on peut trouver l'information!)

En toute généralité, quand on a besoin de documentation, on va consulter un manuel (C'est plus commode d'accès que de la documentation sur écran... et puis, consulter de la documentation "online" crée un job de plus sur un système déjà surchargé). Néanmoins, un programme comme Teach-emacs est bon.

Problème des M-A-J: le plus souvent, elles passent inaperçues(elles n'influencent que des paramètres qu'on n'emploie jamais). Dans le cas contraire, la documentation n'existe pas encore...

Travail en circuit fermé, donc peu intéressé par Bboard, MM, etc.

II.1. AIDE

Au lancement du programme, vous avez le choix entre les trois modes d'accès (date, nom, mot-clé) .

Vous pouvez exprimer votre choix en introduisant le nom du mode d'accès choisi ou une abréviation quelconque de celui-ci. (Les noms sont tels qu'une abréviation réduite à une lettre est discriminante). Un texte d'aide, constitué d'une présentation générale du logiciel, est disponible si vous tapez "?". Après validation par la touche RETURN, l'écran suivant s'affiche :

- Choix par nom :

  Vous introduisez le nom du programme au sujet duquel vous désirez de la documentation. En cas d'introduction d'un "?", vous obtenez l'affichage de la liste des programmes au sujet desquels une fiche d'aide existe. Après validation du choix introduit par la touche RETURN, l'écran contenant la fiche signalétique du programme s'affiche. Après l'affichage de la fiche, si le programme dispose d'un fichier d'aide "nom_du_programme.HLP", vous avez la possibilité de le consulter. Ensuite, le retour s'effectue au menu principal.

- Choix par date :

  Vous devez introduire la date à partir de laquelle vous désirez voir les modifications et les nouveautés. Le programme accepte la date sous toute forme "raisonnable", en anglais ou en français, avec le mois exprimé sous forme numérique ou en toutes lettres, du moment que l'ordre jour-mois-année soit respecté. L'aide éventuelle consiste en un petit texte indiquant les formats admis pour la date. Après validation du choix par la touche RETURN, le programme affiche la liste des programmes dont la documentation a été mise-à-jour après la date introduite. Le retour s'effectue au menu principal.

- Choix par mot-clé :

  L'écran qui apparaît demande l'introduction d'un ou plusieurs mots-clés. Une aide est disponible si nécessaire : elle consiste en une liste des mots-clés existants. Après validation du ou des choix par RETURN, le programme affiche la liste des programmes correspondants au(x) mot(s)-clé(s) introduit(s).

  La recherche des mots_clés correspondants aux mots-clés introduits ne se fait pas sur base d'une comparaison d'égalité stricte. On utilise un algorithme qui considère deux mots-clés comme similaires s'ils ne diffèrent pas par plus de deux caractères consécutifs. L'utilisation de cet algorithme permet de ne pas être arrêté par des différences d'orthographe (p.ex. "mathématique" et "mathématiques", ou "OS" et "O.S."). Si la recherche trouve des mots-clés similaires mais non égaux, le programme vous demande confirmation avant d'accepter le mot-clé trouvé.

Contrairement aux deux cas précédents, le retour après l'affichage de la liste des programmes ne s'effectue pas au menu général : vous avez la possibilité d'affiner la recherche en introduisant des mots-clés supplémentaires, ce qui provoque l'affichage d'une nouvelle liste, sous-ensemble de la première; ce processus pouvant être répeté autant de fois que nécessaire.

Afin de gagner du temps, l'écran intermédiaire peut être court-circuité si, dès le menu général, vous introduisez non seulement le mode de recherche mais aussi une valeur pour les critères souhaités.

II.2. GETEDI

GETEDI est un gestionnaire d'ecran "universel", c'est à dire qu'il fonctionne avec tout terminal ayant un minimum de possibilité de contrôle de curseur.

Le gestionnaire d'écran offre un certain nombre de fonctions que l'on pourrait classer comme suit :

Tout d'abord, on a besoin de **fonctions d'initialisation et de clôture** : elles sont utilisées pour initialiser la gestion d'écran (INITSCR), la clôturer (ENDSCR), pour créer et initialiser de nouvelles fenêtres (NEWWIN et INIT_FIELDS) et pour définir les lignes de message (SET_MSGLINE).

Une fois les fenêtres créées, il faut pouvoir les manipuler : on a donc des **fonctions d'information et de sortie aux fenêtres** : coordonnées minimales, maximales et courantes d'une fenêtre (BEGX, BEGY, MAXX, MAXY, CURX et CURY), contenu (CONTENT et RENDITION), modification du contenu ou de la position courante (MOVE, ADDCHAR et ADDSTR), superposition de deux fenêtres (OVERWRITE et OVERLAY), ajout d'un cadre à une fenêtre (BOX).

Pour la lecture des informations, on dispose d'une **fonction d'entrée** qui permet l'acquisition de caractères dans un contexte de gestion d'écran (READSTR)

Pour voir apparaître les modifications, on a besoin des **fonctions liées au "rafraîchissement" de l'écran** (REFRESH, L_REFRESH, TOUCH et CTRL_L).

Une fois qu'une fenêtre a tenu son rôle, on peut utiliser les **fonctions d'effaçage** : effaçage de l'écran ou de fenêtres (WIPE, ERASE et CLEARALL), effaçage de caractères au terminal (WIPEFIELD).

Pour garder d'une utilisation à l'autre le travail effectué, il existe des **fonctions de sauvegarde des fenêtres** : sauvetage et restauration à partir de fichiers (SAVE et RETRIEVE), effaçage de fenêtres ou de tout un fichier (DEL_WIN et DEL_FICH).

Des **fonctions d'impression** existent, afin de pouvoir garder une trace "écrite" d'un écran : obtention d'une copie papier d'une fenêtre ou de l'écran courant (IMP_ECRAN et HARD_COPY).

Enfin, il existe quelques **fonctions dédiées à la gestion des messages** (MESSAGE, ERREUR, AFF_2MSG, AFF_2ERR, AFF_MSG, CONF_MSG et ECR_MSG).

Pour utiliser le gestionnaire d'écran, voici les opérations à suivre :

- Le système doit connaître quel est le type de votre terminal; le nom logique "TERM:" doit être défini comme un mnémonique de votre terminal (cette opération est normalement effectuée automatiquement lors du "login").

  Les terminaux connus pour le moment sont :

  * Microbeel,
    de Beehive International (mnémonique : BEE1 ou MICROBEE1)
  * Microbee2,
    de Beehive International
    (mnémonique : BEE2 ou MICROBEE2)
  * VT100 de Digital (mnémonique : VT100)
  * GIGI (VK100) de Digital (mnémonique : GIGI)
  * Ann Arbor AMBASSADOR (mnémonique : AMBASSADOR ou AAA)
  * Visual-200 (mnémonique : VISUAL200, V200 ou VIS200)
  * CIT80 de C-ITOH (mnémonique : CIT80)

  Si votre terminal n'est pas dans cette liste, veuiller vous reporter ci-dessous pour connaître la procédure à utiliser pour l'y ajouter.

- Le texte source de votre programme doit inclure, dès après l'instruction "PROGRAM xxx;", les lignes suivantes :

```
include 'sys:string.dec';
include 'sys:cod_ret.dec';
include 'sys:tscr.dec';
include 'sys:getedi.dec';
```

- Votre programme doit être linké avec la librairie "getedi.rel"

Programmation

Avant tout appel à la gestion d'écran, on doit faire appel à la procédure

INITSCR(code_ret : typ_code_ret); si le code retour vaut 'fcdrep', le système de messages n'est pas disponible et les routines de traitement de message ne peuvent être utilisée; pour toute autre valeur [23] 'ok', la routine "ecr_msg" doit être utlilisée, car il y a eu une erreur dans l'initialisation de la gestion d'écran.

A la fin du programme, il faut appeler la procédure ENDSCR pour clôturer la gestion d'écran.

Voici maintenant une description des différentes fonctions :

Fonctions d'initialisation et de clôture

PROCEDURE initscr(VAR code_ret : typ_code_ret);

Initialise les différentes choses nécessaires pour la gestion d'écran . Le code de retour vaut 'ok' en cas de réussite; En cas d'erreur,l'initialistion n'est pas faite; (en particulier, il n'est pas possible d'employer des routines faisant appel à la gestion d'écran pour afficher les messages d'erreur

Valeur des codes-retour :
ok
fcdrep : fichier des codes-retour n'existe pas
fecrep : fichier de description des ecrans n'existe pas
tnd : terminal non défini (pas de nom logique TERM:)
tinc : terminal inconnu (pas de description)

PROCEDURE endscr;

Termine la gestion d'écran et imprime le fichier d'impression

PROCEDURE init_fields(VAR curwin : window;all_screen : BOOLEAN);

Initialise les différentes composantes de la représentation d'une fenêtre; Si le booléen "all-screen" est vrai, l'initialisation se fait sur toute la représentation physique de la fenêtre (qui n'est effectivement utilisée que dans le cas d'une fenêtre de taille maximale).Sinon, l'initialisation ne se fait que sur la partie de la représentation physique qui sert effectivement à représenter la fenêtre

FUNCTION newwin(homex,homey,edgex,edgey : INTEGER):window;

Crée la représentation d'une fenêtre débutant à "homex", "homey", dont le coin inférieur droit est placé en "edgex","edgey" , et retourne un pointeur vers cette représentation

```
PROCEDURE set_msgline(i :INTEGER);
```

Définit les 2 lignes de l'écran où seront affichés les messages ("i" : première ligne de messages)

Fonctions d'information et de sortie aux fenêtres

```
FUNCTION begx(curwin : window) : INTEGER;
FUNCTION begy(curwin : window) : INTEGER;
FUNCTION maxx(curwin : window) : INTEGER;
FUNCTION maxy(curwin : window) : INTEGER;
FUNCTION curx(curwin : window) : INTEGER;
FUNCTION cury(curwin : window) : INTEGER;

FUNCTION content(curwin : window;x,y : INTEGER):CHAR;

FUNCTION rendition(curwin : window;x,y : INTEGER):typ_rendition;
```

Ces fonctions permettent d'acquérir la position minimale, maximale, et courante d'une fenêtre, ainsi que le contenu et le mode d'affichage d'une position.

```
PROCEDURE move(VAR curwin : window;x,y : INTEGER);
```

Modifie la position courante de "curwin" en "x","y"

```
PROCEDURE addchar(VAR curwin : window; c: CHAR; rend : typ_rendition);
```

Ajoute le caractère "c",en mode d'affichage "rend",à la position courante de la fenêtre "curwin". La position courante en "y" est avancée d'une position.

```
PROCEDURE addstr(VAR curwin : window;
VAR string : PACKED ARRAY [m..n:INTEGER ] OF CHAR;
rendition : typ_rendition );
```

Ajoute à l'écran "curwin" la chaîne de caractères "string" (en ignorant les 'trailing blanks'), à partir de la position courante de l'écran.
Si la chaine est trop longue, elle est tronquée. Le caractère dans le coin inférieur droit est forcé à blanc pour éviter un 'scrolling' lors d'un refresh ulterieur.
Les caractères de tabulation sont convertis en blancs.

```
PROCEDURE overwrite(VAR win1,win2 : window);
```

Réécrit le contenu de la fenêtre "win1" sur la fenêtre "win2" de façon destructive : les blancs sur "win1" deviennent des blancs sur "win2"

```
PROCEDURE overlay(VAR win1,win2 : window);
```

Réécrit le contenu de la fenêtre "win1" sur la fenêtre "win2" de façon non-destructive : à l'emplacement des blancs de "win1", l'ancien contenu de "win2" subsiste

```
PROCEDURE box(VAR curwin : window; cx : CHAR; cy : CHAR);
```

Trace un cadre sur les bords de la fenêtre "curwin", en employant le caractère "cx" pour la dimension des x, et le caractere "cy" pour la dimension des y .

Fonction d'entrée

```
PROCEDURE readstr(x,y : INTEGER;
                  n_char_to_get : INTEGER;
                  VAR result : PACKED ARRAY [m..n:INTEGER]
                  OF CHAR;
                  VAR count : INTEGER;
                  break_set : char_set;
                  rendition : typ_rendition;
                  VAR code_ret : typ_code_ret);
```

Garnit le tableau de caractères "result" avec des caractères pris au terminal, à partir de la position courante de l'écran "curwin", SANS MODIFIER CET ECRAN D'AUCUNE FACON, et s'arrété lorsque
- "n_char_to_get" caractères ont été lus
- un caractère repris dans "break_set" est rencontré
- le tableau "result" est rempli

"count" contient le nombre de caractères effectivement lus;
En cours d'introduction, il est possible d'utiliser les touches "delete" et Control-U

Une facilité est prévue pour une aide éventuelle :
si un '?' est introduit comme premier caractère, la lecture s'arréte et le code de retour est positionné pour indiquer cette éventualité.

Valeurs du code_retour :
ok
inter : le premier caractère lu est un "?"

Fonctions liées au rafraîchissement

```
PROCEDURE refresh(VAR curwin : window);
```

Synchronise l'écran physique avec le contenu de la fenêtre "curwin" tel que le connaît le gestionnaire d'écran

```
PROCEDURE l_refresh(x,y : INTEGER);
```

Synchronise la ligne "x", à partir de la position "y", avec cette partie de CURSCR

```
PROCEDURE touch(VAR curwin : window);
```

Marque toutes les positions de l'écran "curwin" comme ayant été modifiées ; ceci implique qu'au prochain refresh, tout l'écran sera réécrit

```
PROCEDURE ctrl_l;
```

Redessine l'écran tel que le gestionnaire d'écran le connaît

Fonctions d'effaçage

```
PROCEDURE wipe(VAR curwin : window);
```

Provoque un effaçage de l'écran du terminal lors du prochain 'refresh'. Cet effaçage a lieu sur l'étendue de la fenêtre "curwin". Le contenu de l'écran n'est pas affecté

```
PROCEDURE erase(VAR curwin : window);
```

Provoque un effaçage du contenu de la fenêtre "curwin". L'écran du terminal n'est pas affecté

```
PROCEDURE clearall(VAR curwin : window);
```

Provoque un effaçage du contenu de la fenêtre "curwin" et de l'écran du terminal(effectif au prochain 'refresh')

```
PROCEDURE wipefield(VAR curwin : window;nb_car : INTEGER);
```

Efface "nb_car" caractères sur l'écran du terminal,sur la ligne "curx" de "curwin",à partir de la position "cury", SANS AFFECTER L'ECRAN LOGIQUE.

Fonctions liées à la sauvegarde des fenêtres

```
PROCEDURE save(curwin : window; nom_win : str10;
               VAR nom_fich : PACKED ARRAY [m..n:INTEGER] OF CHAR;
               VAR code_ret : typ_code_ret );
```

Sauve le contenu de la fenêtre "curwin" , sous le nom "nom_win" dans le fichier nommé "nom_fich". Le code de retour "code_ret" vaut 'ok' si tout s'est bien passé, ou un code signalant l'anomalie rencontrée sinon.

Valeur des codes retour :
ok
wed : une fenêtre ayant nom donné en argument existe déjà

```
PROCEDURE retrieve(VAR curwin : window;nom_win : str10;
                   VAR nom_fich : PACKED ARRAY [m..n:INTEGER] OF CHAR ;
                   VAR code_ret : typ_code_ret);
```

Crée la fenêtre "curwin" et la garnit avec le contenu de la fenêtre enregistrée sous le nom "nom_win" dans le fichier nommé "nom_fich".
Le code de retour "code_ret" vaut 'ok' si tout s'est bien passé, ou un code signalant l'anomalie rencontrée dans le cas contraire.
Si une fenêtre "curwin" existait déjà, son contenu est perdu.

Valeurs du code retour :
ok
fne : fichier (des fenêtres) n'existe pas
ept : écran (a restaurer) pas trouvé

```
PROCEDURE del_fich(VAR nom_fich : PACKED ARRAY [m..n:INTEGER] OF CHAR;
                   VAR code_ret : typ_code_ret);
```

Efface le fichier de nom "nom_fich".
Le code de retour "code_ret" vaut 'ok' si tout s'est bien passé, ou un code signalant l'anomalie rencontrée dans le cas contraire.

Valeurs du code retour :
ok
fne : fichier (à effacer) n'existe pas

```
PROCEDURE del_win(nom_win : str10;
VAR nom_fich : PACKED ARRAY [m..n:INTEGER] OF CHAR ;
VAR code_ret : typ_code_ret);
```

Efface la fenêtre de nom "nom_win" du fichier nommé "nom_fich". Le code de retour "code_ret" vaut 'ok' si tout s'est bien passe, ou un code signalant l'anomalie rencontrée dans le cas contraire.

Valeurs du code_retour :
ok
fne : fichier (dont on veut effacer une fenêtre) n'existe pas
wne : fenêtre (que l'on veut effacer) n'existe pas

Fonctions d'impression

```
PROCEDURE imp_ecran(VAR curwin : window;cadre : BOOLEAN);
```

Envoie le contenu de l'écran "curwin" vers un fichier d'impression, qui sert de SPOOLER et sera envoyé à l'impression physique à la sortie du gestionnaire d'écran.
Si le booléen "cadre" est TRUE, l'écran sera 'encadré par une numérotation des lignes et colonnes.

```
PROCEDURE hard_copy(cadre : BOOLEAN);
```

Réalise une copie sur le fichier d'impression de l'écran physique du terminal tel que le connaît le gestionnaire d'écran

Fonctions de traitement des messages

```
PROCEDURE message(VAR texte_1 : PACKED ARRAY [m..n:INTEGER] OF CHAR ;
                  VAR texte_2 : PACKED ARRAY [p..q:INTEGER] OF CHAR ;
                  VAR c: CHAR );
```

Affiche un message de 2 lignes, en mode normal, sans modifier le contenu de l'écran courant.
Le message reste affiché jusqu'à l'introduction d'un caractère quelconque, qui est par après disponible en "c" è

```
PROCEDURE erreur(VAR texte_1 : PACKED ARRAY [m..n:INTEGER] OF
                 CHAR ;
                 VAR texte_2 : PACKED ARRAY [p..q:INTEGER] OF
                 CHAR ;
                 VAR c: CHAR );
```

Affiche un message d'erreur de 2 lignes, avec la première ligne en vidéo inverse et la seconde en mode normal, et actionne le beeper du terminal.
Le message reste affiché jusqu'à l'introduction d'un caractère quelconque, qui est par après disponible en "c"

```
PROCEDURE aff_2msg(msg_1, msg_2 : typ_code_ret; VAR c : CHAR
```

Affiche les messages correspondants aux codes-retour "msg_1" et "msg_2" sur les lignes destinées aux messages.
Garde les messages affichés jusqu'à l'introduction d'un caractère, disponible en "c"

PROCEDURE aff_2err(msg_1, msg_2 : typ_code_ret;VAR c : CHAR);

Affiche les messages correspondants aux codes-retour "msg_1" et "msg_2" sur les lignes destinées aux messages, le premier en vidéo inverse et le second en mode normal.
Garde les messages affichés jusqu'à l'introduction d'un caractère, disponible en "c"

PROCEDURE aff_msg(msg : typ_code_ret);

Affiche le message correspondant au code retour "code_ret" sur la ligne destinée aux messages, et un message approprié à la nature du premier sur la deuxième ligne.
Garde le message affiché jusqu'à l'introduction d'un caractère

FUNCTION conf_msg(msg : typ_code_ret): BOOLEAN;

Affiche le message correspondant au code retour "msg" sur la première ligne destinée aux messages,ainsi qu'un message de demande de confirmation sur la seconde ligne.
Garde le message affiché jusqu'à l'introduction d'un caractère.
Retourne TRUE si ce caractère est 'o' ou 'O' ou 'y' ou 'Y' , FALSE sinon .

PROCEDURE ecr_msg(msg : typ_code_ret);

Affiche les messages correspondant au code retour "code_ret" sans faire appel à la gestion d'écran. Cette procédure peut être utilisée pour afficher les messages lorsque l'initialisation ne réussit pas ou hors d'un contexte de gestion d'écran.

A l'exception de trois fonctions, les procédures de gestion d'écran physiques ne doivent pas être appelées par un programme d'application.

Voici ces trois fonctions :

- FUNCTION xphysx, FUNCTION xphysy : Donnent les dimensions de l'écran physique du terminal sur lequel tourne l'application;
- PROCEDURE xcls : provoque l'effaçage de l'écran physique.

Description d'un nouveau terminal par MKDESC

Pour qu'un terminal puisse être utilisé avec GETEDI, il doit avoir les possibilités suivantes :

- Contrôle de curseur absolu et relatif
- Effacement depuis le curseur jusqu'à la fin de la ligne ou de l'écran, effacement de tout l'écran

Facultativement :

- Possibilités d'affichage vidéo inverse, en surintensité, en clignotant, en souligné

Les terminaux peuvent avoir besoin de séquences de caractères d'initialisation et de clôture.

Pour introduire la description d'un nouveau terminal, il faut éditer le programme MKDESC, y introduire les caractéristiques du nouveau terminal (on peut prendre comme exemple la description d'un terminal qui s'y trouve déjà), et enfin, réexécuter le programme afin de créer un nouveau fichier de description des terminaux.

II.3. INTRO

INTRO permet différentes opérations sur les "fiches signalétiques" des programmes : introduction, mise-à-jour, suppression.

Pour l'introduction, vous devez d'abord introduire le nom du programme à documenter : ceci permet de vérifier qu'un programme ayant un tel nom n'existe pas encore. S'il existe, le programme affiche la fiche signalétique correspondante et on annule la suite des opérations. Dans le cas contraire, vous êtes invité à introduire : la fonction du programme (en une ligne), la date (avec une date par défaut qui est la date du jour), les mots-clés (maximum 10), et les moyens d'obtenir de la documentation supplémentaire (trois lignes maximum). En ce qui concerne les mots-clés, on vérifie l'existence de doubles ou de "quasi-doubles" avec un algorithme de "similarité" identique à celui employé pour l'exploitation en recherche de la collection des mots-clés.

Pour la mise-à-jour, la procédure est similaire : introduction du nom du programme, annulation si la fiche n'existe pas, introduction des autres renseignements, avec vérification des doubles pour les mots-clés. La principale différence par rapport à l'ajout est que pour chaque champ, une valeur par défaut est fournie, consistant en la valeur ancienne de ce champ de la fiche (à l'exception de la date, où le défaut est la date du jour).

Pour la suppression, vous ne devez introduire que le nom du programme dont on veut supprimer la fiche. Si elle n'existe pas, on le signale et on annule les opérations. Sinon, on affiche le contenu de la fiche à supprimer. Après confirmation, le programme effectue les opérations nécessaires à la suppression.

II.4. SCREEN

Le mini-éditeur SCREEN permet de créer et de modifier les fenêtres destinés à GETEDI.

Les trois opérations de base sont ici aussi la création, la modification et la suppression.

Dans les trois cas, les renseignements à introduire sont le nom du fichier des fenêtres et le nom de la fenêtre (pour le nom du fichiers des fenêtres, le programme fournit un défaut, s'il existe : le nom du fichier des fenêtres précédent).

Pour la création, le programme vous demande en plus les dimensions de la fenêtre à créer (par défaut, ce sont les dimensions de l'écran du terminal physique sur lequel tourne l'éditeur).

Pour la suppression, vous avez le choix entre la suppression de tous les écrans ou celle d'un écran particulier.

Lors de la création et de la mise-à-jour, les principales fonctions proposées sont : les mouvements de curseur, l'ajout de texte, la suppression (réalisée par ajout de texte blanc) et le tracé d'un cadre à l'écran.

III. LAYOUT DES ÉCRANS D'AIDE

IV. SOURCES

Voir listings.

TABLE DES MATIERES

FACULTES UNIVERSITAIRES NOTRE DAME DE LA PAIX, NAMUR

INSTITUT D'INFORMATIQUE

ANNEXES DU MEMOIRE DE K. BOKOR

"AIDE : UN LOGICIEL DE DOCUMENTATION EN LIGNE INTERACTIF "

ANNEE ACADEMIQUE 1983 - 1984

Ces annexes reprennent

- les "layout" des ecrans du logiciel AIDE
- les sources des programmes

(accompagne de la liste des codes retour employes et des messages associes)

FENETRE D'AFFICHAGE D'UNE FICHE SIGNALETIQUE

```
          1         2         3         4         5         6         7         8
 12345678901234567890123456789012345678901234567890123456789012345678901234567890
 1|                                                                              | 1
 2|                                                                              | 2
 3|                                                                              | 3
 4|                                                                              | 4
 5|Voici les renseignements de base sur le programme demande :                   | 5
 6|                                                                              | 6
 7|Nom :                                                                         | 7
 8|                                                                              | 8
 9|Fonction :                                                                    | 9
10|                                                                              |10
11|                                                                              |11
12|Date :                                                                        |12
13|                                                                              |13
14|Autre documentation :                                                         |14
15|                                                                              |15
16|                                                                              |16
17|                                                                              |17
18|                                                                              |18
19|Mots-cles :                                                                   |19
20|                                                                              |20
21|                                                                              |21
22|                                                                              |22
23|                                                                              |23
24|                                                                              |24
25|                                                                              |25
26|                                                                              |26
27|                                                                              |27
28|                                                                              |28
29|                                                                              |29
30|                                                                              |30
 12345678901234567890123456789012345678901234567890123456789012345678901234567890
          1         2         3         4         5         6         7         8
```

FENETRE D'AIDE POUR L'INTRODUCTION DE LA DATE

```
          1         2         3         4         5         6         7         8
01234567890123456789012345678901234567890123456789012345678901234567890123456789
 1|                                                                              | 1
 2|                                                                              | 2
 3|                                                                              | 3
 4|                                                                              | 4
 5|                                                                              | 5
 6|  La date peut etre introduite sous toute forme respectant la contrainte :    | 6
 7|                                                                              | 7
 8|              JOUR avant MOIS avant ANNEE.                                    | 8
 9|                                                                              | 9
10|                                                                              |10
11|         P.ex : 19 7 60                                                       |11
12|                 19 juillet 1960                                              |12
13|                 19/7/60                                                      |13
14|                 19-jul-60                                                    |14
15|                                                                              |15
16|                   etc.                                                       |16
17|                                                                              |17
18|                                                                              |18
19|                                                                              |19
20|                                                                              |20
21|                                                                              |21
22|                                                                              |22
23|                                                                              |23
24|                                                                              |24
25|                                                                              |25
26|                                                                              |26
27|                                                                              |27
28|                                                                              |28
29|                                                                              |29
30|                                                                              |30
01234567890123456789012345678901234567890123456789012345678901234567890123456789
          1         2         3         4         5         6         7         8
```

FENETRE DE DEMANDE DE NOM DE PROGRAMME

```
          1         2         3         4         5         6         7         8
  12345678901234567890123456789012345678901234567890123456789012345678901234567890
 1|Sur quel programme desirez-vous de l'aide ?                                      | 1
 2|                                                                                 | 2
 3|---------------------------------------------------------------------------------| 3
 4|                                                                                 | 4
 5|                                                                                 | 5
 6|                                                                                 | 6
 7|                                                                                 | 7
 8|                                                                                 | 8
 9|                                                                                 | 9
10|                                                                                 |10
11|                                                                                 |11
12|                                                                                 |12
13|                                                                                 |13
14|                                                                                 |14
15|                                                                                 |15
16|                                                                                 |16
17|                                                                                 |17
18|                                                                                 |18
19|                                                                                 |19
20|                                                                                 |20
21|                                                                                 |21
22|                                                                                 |22
23|                                                                                 |23
24|                                                                                 |24
25|                                                                                 |25
26|                                                                                 |26
27|                                                                                 |27
28|                                                                                 |28
29|                                                                                 |29
30|                                                                                 |30
  12345678901234567890123456789012345678901234567890123456789012345678901234567890
          1         2         3         4         5         6         7         8
```

FENETRE DE DEMANDE DES MOTS-CLES

```
          1         2         3         4         5         6         7         8
   12345678901234567890123456789012345678901234567890123456789012345678901234567890
 1|A propos de quel(s) mot(s)-cle(s) desirez-vous de l'aide ?                        | 1
 2|                                                                                  | 2
 3|----------------------------------------------------------------------------------| 3
 4|                                                                                  | 4
 5|                                                                                  | 5
 6|                                                                                  | 6
 7|                                                                                  | 7
 8|                                                                                  | 8
 9|                                                                                  | 9
10|                                                                                  |10
11|                                                                                  |11
12|                                                                                  |12
13|                                                                                  |13
14|                                                                                  |14
15|                                                                                  |15
16|                                                                                  |16
17|                                                                                  |17
18|                                                                                  |18
19|                                                                                  |19
20|                                                                                  |20
21|                                                                                  |21
22|                                                                                  |22
23|                                                                                  |23
24|                                                                                  |24
25|                                                                                  |25
26|                                                                                  |26
27|                                                                                  |27
28|                                                                                  |28
29|                                                                                  |29
30|                                                                                  |30
   12345678901234567890123456789012345678901234567890123456789012345678901234567890
          1         2         3         4         5         6         7         8
```

FENETRE DE DEMANDE D'INTRODUCTION DE DATE

```
          1         2         3         4         5         6         7         8
 12345678901234567890123456789012345678901234567890123456789012345678901234567890
 1|A partir de quelle date desirez-vous voir les modifications ?                    | 1
 2|                                                                                 | 2
 3|---------------------------------------------------------------------------------| 3
 4|                                                                                 | 4
 5|                                                                                 | 5
 6|                                                                                 | 6
 7|                                                                                 | 7
 8|                                                                                 | 8
 9|                                                                                 | 9
10|                                                                                 |10
11|                                                                                 |11
12|                                                                                 |12
13|                                                                                 |13
14|                                                                                 |14
15|                                                                                 |15
16|                                                                                 |16
17|                                                                                 |17
18|                                                                                 |18
19|                                                                                 |19
20|                                                                                 |20
21|                                                                                 |21
22|                                                                                 |22
23|                                                                                 |23
24|                                                                                 |24
25|                                                                                 |25
26|                                                                                 |26
27|                                                                                 |27
28|                                                                                 |28
29|                                                                                 |29
30|                                                                                 |30
 12345678901234567890123456789012345678901234567890123456789012345678901234567890
          1         2         3         4         5         6         7         8
```

FENETRE DE PRESENTATION DU "MENU GENERAL"

```
          1         2         3         4         5         6         7         8
01234567890123456789012345678901234567890123456789012345678901234567890123456789
 1|                          ********                                           | 1
 2|                          * AIDE *                                           | 2
 3|                          ********                                           | 3
 4|                                                                             | 4
 5|                                                                             | 5
 6|                                                                             | 6
 7|Quel genre d'aide souhaitez-vous ?                                           | 7
 8|                                                                             | 8
 9|  - NOM : Aide sur un programme dont vous donnez le nom                      | 9
10|                                                                             |10
11|  - MOTS-CLES : Liste des programmes correspondants a une liste de mots-cles |11
12|                                                                             |12
13|  - DATE : Liste des nouveautes et modifications depuis une date donnee      |13
14|                                                                             |14
15|    ( QUITTER pour terminer )                                                |15
16|                                                                             |16
17|                                                                             |17
18|                                                                             |18
19|                                                                             |19
20|                                                                             |20
21|          Votre choix :                                                      |21
22|                                                                             |22
23|                                                                             |23
24|                                                                             |24
25|                                                                             |25
26|                                                                             |26
27|                                                                             |27
28|                                                                             |28
29|                                                                             |29
30|                                                                             |30
01234567890123456789012345678901234567890123456789012345678901234567890123456789
          1         2         3         4         5         6         7         8
```

Liste des codes-retours et des messages associes

No	Code	Message associe
1	ok	OK
2	deb_cod_avec_msg	(* debut des messages correspondants aux codes-retour *)
3	fecreb	Le fichier de description des terminaux n'existe pas.
4	tnd	Veuillez definir TERM: comme votre terminal
5	tinc	Ce type de terminal n'est pas connu
6	fcdreb	Pas de fichier de ce nom
7	fne	Une fenetre de ce nom existe deja
8	wed	Ecran pas trouve
9	ebt	Pas de fenetre de ce nom
10	wne	La zone "jour" doit etre numerique
11	jnum	La longueur de la zone "jour" doit etre <= 2
12	li2	La longueur de la zone "mois" doit etre <= 2
13	lm2	La longueur de la zone "mois" doit etre <= 10
14	lm10	Mois inconnu
15	mnc	La zone "annee" doit etre numerique
16	anum	Le mois doit etre > 1 et < 12
17	rmo	Le jour doit etre <= 31
18	ri31	Le jour doit etre <= 30
19	ri30	Le jour doit etre <= 29
20	ri29	Le jour doit etre <= 28
21	ri28	L'annee demandee doit etre <= a l'annee courante
22	anin	Le mois demande doit etre <= au mois courant
23	moin	Le jour demande doit etre <= au jour courant
24	join	Pas d'ecran precedent
25	ddf	Pas d'ecran suivant
26	fdf	Commande inconnue
27	cin	Erreur imprevue.Veuillez vous adressez a une personne competente du CC.
28	callguru	Aaargh! Quelque chose ne va pas ! Message inconnu.
29	minc	Bizarre...ce n'est pas un message demandant confirmation...
30	paconf	(* fin des messages d'erreur *)
31	fin_err	Fichier efface
32	fef	Sauvetage effectue
33	sef	Rappel effectue
34	ref	Fenetre effacee
35	wef	(repondez par 'o' ou 'n')
36	dco	(tapez un caractere quelconque pour continuer)
37	mic	Pas de programme correspondant
38	ppc	Commande vide...
39	comvid	Impression effectuee
40	impok	Quel mode d'affichage ?
41	qrend	Impression avec cadre ou sans ?
42	qcadr	Choix : (n)ormal, (i)nverse, (c)lignotant, (g)ras, (s)ouligne
43	chrend	Repondez 'c' pour avoir un cadre, ou un autre caractere pour continuer
44	chcadr	(* fin des messages generaux *)
45	fin_msg	Desirez-vous voir le fichier "help" du programme demande ?
46	chlp	(* fin des messages demandant confirmation *)
47	fin_conf	
48	inter	
49	termin	
50	rien	

```
'ROGRAM aide;

{ copyright 1984 by Charles BOKOR and 'Centre de Calcul' of Facultes
universitaires Notre Dame de la Paix,Namur,Belgium }

{ Ce programme a ete ecrit et mis au point en mars/avril 1984 par
Charles BOKOR, dans le cadre d'un memoire de fin d'etudes de licence
et maitrise en informatique visant a creer un systeme de
documentation on line a interface utilisateur evolue.
Ce qui suit constitue le programme de base et les routines du logiciel
de documentation qui a ete developpe. Ce logiciel utilise les services
du gestionnaires d'ecran GETEDI.}
```

```
nclude 'sys:string.dec';        { declarations de types str** comme
                                  etant des PACKED ARRAY [1..**] OF CHAR
                                  et de constantes bl** (strings de
                                                  blancs equivalents) }
nclude 'sys:elems.dec';         { declarations des elements des listes
                                  diverses, des fichiers de sauvetage
                                  de ces listes, et du fichier des
                                  renseignments du logiciel de docu }
nclude 'sys:cod_ret.dec';       { definitions des constantes et types pour l'
                                  affichages des messages d'erreur ou de
                                  confirmation }
nclude 'sys:utilit.dec';        { declaration de quelques procedures et
                                  fonctions 'utilitaires' }
nclude 'sys:tscr.dec';          { definitions des constantes,types et
                                  procedures necessaires pour la gestion
                                  de l'ecran du terminal physique }
nclude 'sys:getedi.dec';        { definitions des constantes,types et
                                  procedures necessaires pour la gestion
                                  d'ecran logique }

ABEL 99,100;

ONST sepa = 4;
     nom_fich_fenetre = 'sys:aid.screen';
     nom_fich_mc = 'sys:fmc.hlp';
     nom_fich_date = 'sys:fdate.hlp';
     nom_fich_nom = 'sys:fnom.hlp';
     nom_fich_aide = 'sys:fhlp.hlp';
     nom_meta_fich_aide = 'sys:aide.hlp';

YPE typ_commande = RECORD
       CASE intitule : CHAR OF
          'M' : (ldem : d_el_ldem);
          'D' : (date_in : str10);
          'N' : (nom_fich : str20);
    END;

AR lmc : d_el_lmc;              { listes }
   lnom : d_el_lnom;
   ldate : d_el_ldate;
   lres : d_el_lres;

   fnom : f_el_nom;             { fichiers de sauvetage des listes }
   fmc : f_el_mc;
   fdate : f_el_date;
   fhlp : f_el_hlp;


   code_ret : typ_code_ret;
   i : INTEGER;

   cde : typ_commande;

   w_menu_gen,w_dem_date,w_dem_mc,w_dem_nom,w_aide_date : window;
   w_aff_nom,w_squel_nom,w_mc_sim,w_aff_dem : window;

   ligne_blanche : PACKED ARRAY [1..lim_col] OF CHAR;
```

procedures de traitement des listes :

fonction X_cree :
IN :
valeur de la fonction : valeur initiale du designateur d'une liste de X
semantique : cree une liste de X

fonction X_suivant :
IN : liste : designateur de la liste de X
element : designateur de l'element dont on cherche le suivant
valeur de la fonction : designateur vers l'element suivant
semantique : si "element" est vide, fournit le premier de la liste de X
"liste"
sinon, fournit l'element suivant "element" dans "liste"

procedure X_content :
IN : element : designateur d'un element de la liste des X
OUT : contenu : partie "content" du contenu de l'element "element"
semantique : si "element" n'est pas vide, fournit la valeur de la partie
'content' du contenu de celui-ci;
sinon, fournit un string de valeur ASCII maximale.

procedure X_adresse :
IN : element : designateur d'un element de la liste des X
OUT : adresse : partie 'adresse ' du contenu de l'element "element"
semantique : si "element" n'est pas vide, fournit la valeur de la partie
'adresse ' du contenu de celui-ci;
sinon, fournit la valeur MAXINT

procedure set_X_content :
IN : element : designateur d'un element de la liste des X
contenu : valeur que l'on souhaite affecter a l'element "element"
OUT :
semantique : assigne la valeur "contenu" a la partie 'content' de
l'element "element".

procedure X_ins :
IN : tete : designateur de la liste des X
contenu : contenu de l'element a inserer
prec : designateur de l'element apres lequel on souhaite inserer
OUT : liste de X modifiee
semantique : insere un element de contenu "contenu" dans la liste
de 'X' designee par "tete",
apres l'element designe par "prec" si
"prec" n'est pas vide,
en debut de liste si "prec" est vide.

procedure X_discard :
IN : liste : designateur de la liste des X
semantique : detruit la liste dont le designateur est donne en
argument. Apres cette procedure, la liste vaut VIDE.

avec X valant lmc,ldem,lres,ldate,lnom.

```
{-------------------------------------------------------------------------}

FUNCTION lmc_cree : d_el_lmc;
BEGIN
  lmc_cree := NIL;
END;

FUNCTION ldem_cree : d_el_ldem;
BEGIN
  ldem_cree := NIL;
END;

FUNCTION lres_cree : d_el_lres;
BEGIN
  lres_cree := NIL;
END;

FUNCTION ldate_cree : d_el_ldate;
BEGIN
  ldate_cree := NIL;
END;

FUNCTION lnom_cree : d_el_lnom;
BEGIN
  lnom_cree := NIL;
END;

{-------------------------------------------------------------------------}

FUNCTION lmc_suivant(liste : d_el_lmc;element : d_el_lmc) : d_el_lmc;
BEGIN
  IF element = NIL THEN lmc_suivant := liste
                   ELSE lmc_suivant := element^.next;
END;

FUNCTION ldem_suivant(liste : d_el_ldem;element : d_el_ldem) : d_el_ldem;
BEGIN
  IF element = NIL THEN ldem_suivant := liste
                   ELSE ldem_suivant := element^.next;
END;

FUNCTION lres_suivant(liste : d_el_lres;element : d_el_lres) : d_el_lres;
BEGIN
  IF element = NIL THEN lres_suivant := liste
                   ELSE lres_suivant := element^.next;
END;

FUNCTION ldate_suivant(liste : d_el_ldate;element : d_el_ldate) : d_el_ldate;
BEGIN
  IF element = NIL THEN ldate_suivant := liste
                   ELSE ldate_suivant := element^.next;
END;

FUNCTION lnom_suivant(liste : d_el_lnom;element : d_el_lnom) : d_el_lnom;
BEGIN
  IF element = NIL THEN lnom_suivant := liste
                   ELSE lnom_suivant := element^.next;
END;
```

```
-------------------------------------------------------------------------------}

 les procedures qui suivent ne sont des procedures que parce que ce %^$%#%$
 de PASCAL n'admet pas des tableaux de caracteres comme resultat d'une
 fonction }

ROCEDURE lmc_content(element : d_el_lmc;VAR contenu : str20);
EGIN
  IF element <> NIL THEN contenu := element^.content
                    ELSE contenu := maxstr20;
ND;

ROCEDURE ldem_content(element : d_el_ldem;VAR contenu : str20);
EGIN
  IF element <> NIL THEN contenu := element^.content
                    ELSE contenu := maxstr20;
ND;

ROCEDURE lres_content(element : d_el_lres;VAR contenu : INTEGER);
pourrait etre une fonction. Sous forme de procedure dans un souci
d'uniformite }
EGIN
  IF element <> NIL THEN contenu := element^.content
                    ELSE contenu := MAXINT;
ND;

ROCEDURE ldate_content(element : d_el_ldate;VAR contenu : str10);
EGIN
  IF element <> NIL THEN contenu := element^.content
                    ELSE contenu := maxstr10;
ND;

ROCEDURE lnom_content(element : d_el_lnom;VAR contenu : str20);
EGIN
  IF element <> NIL THEN contenu := element^.content
                    ELSE contenu := maxstr20;
ND;

-------------------------------------------------------------------------------}

ROCEDURE lmc_adresse(element : d_el_lmc;VAR adresse : INTEGER);
EGIN
  IF element <> NIL THEN adresse := element^.adresse
                    ELSE adresse := MAXINT;
ND;

ROCEDURE ldate_adresse(element : d_el_ldate;VAR adresse : INTEGER);
EGIN
  IF element <> NIL THEN adresse := element^.adresse
                    ELSE adresse := MAXINT;
ND;

ROCEDURE lnom_adresse(element : d_el_lnom;VAR adresse : INTEGER);
EGIN
  IF element <> NIL THEN adresse := element^.adresse
                    ELSE adresse := MAXINT;
ND;

-------------------------------------------------------------------------------}
```

```
ROCEDURE set_ldem_content(element : d_el_ldem;contenu : str20);
EGIN
  IF element <> NIL THEN element^.content := contenu;
ND;

----------------------------------------------------------------------------}

ROCEDURE lmc_ins(VAR tete : d_el_lmc; contenu : el_mc;
                     VAR prec : d_el_lmc);

AR nouv_mc : d_el_lmc;

EGIN
  NEW(nouv_mc);
  nouv_mc^.content := contenu.content;
  nouv_mc^.adresse := contenu.adresse;
  IF prec = NIL THEN
     BEGIN
        nouv_mc^.next := tete;
        tete := nouv_mc;
     END
  ELSE
     BEGIN
        nouv_mc^.next := prec^.next;
        prec^.next := nouv_mc;
     END;
ND;

----------------------------------------------------------------------------}

ROCEDURE ldem_ins(VAR tete : d_el_ldem; contenu : str20;
                      VAR prec : d_el_ldem);

AR nouv_dem : d_el_ldem;

EGIN
  NEW(nouv_dem);
  nouv_dem^.content := contenu;
  IF prec = NIL THEN
     BEGIN
        nouv_dem^.next := tete;
        tete := nouv_dem;
     END
  ELSE
     BEGIN
        nouv_dem^.next := prec^.next;
        prec^.next := nouv_dem;
     END;
ND;

----------------------------------------------------------------------------}

ROCEDURE lres_ins(VAR tete : d_el_lres; contenu : INTEGER ;
                      VAR prec : d_el_lres);

AR nouv_res : d_el_lres;

EGIN
  NEW(nouv_res);
  nouv_res^.content := contenu;
```

```
  IF prec = NIL THEN
     BEGIN
        nouv_res^.next := tete;
        tete := nouv_res;
     END
  ELSE
     BEGIN
        nouv_res^.next := prec^.next;
        prec^.next := nouv_res;
     END;
ND;

----------------------------------------------------------------------------}

ROCEDURE ldate_ins(VAR tete : d_el_ldate; contenu : el_date;
                       VAR prec : d_el_ldate);

AR nouv_date : d_el_ldate;

EGIN
  NEW(nouv_date);
  nouv_date^.content := contenu.content;
  nouv_date^.adresse := contenu.adresse;
  IF prec = NIL THEN
     BEGIN
        nouv_date^.next := tete;
        tete := nouv_date;
     END
  ELSE
     BEGIN
        nouv_date^.next := prec^.next;
        prec^.next := nouv_date;
     END;
ND;

----------------------------------------------------------------------------}

ROCEDURE lnom_ins(VAR tete : d_el_lnom; contenu : el_nom;
                      VAR prec : d_el_lnom);

AR nouv_nom : d_el_lnom;

EGIN
  NEW(nouv_nom);
  nouv_nom^.content := contenu.content;
  nouv_nom^.adresse := contenu.adresse;
  IF prec = NIL THEN
     BEGIN
        nouv_nom^.next := tete;
        tete := nouv_nom;
     END
  ELSE
     BEGIN
        nouv_nom^.next := prec^.next;
        prec^.next := nouv_nom;
     END;
ND;

============================================================================}
```

```
ROCEDURE ldem_discard(VAR ldem : d_el_ldem);

AR temp,suiv : d_el_ldem;

EGIN
  temp := ldem;
  WHILE temp <> NIL DO
     BEGIN
        suiv := temp^.next;
        DISPOSE(temp);
        temp := suiv;
     END;
  ldem := NIL;
ND;

----------------------------------------------------------------------------}

ROCEDURE lres_discard(VAR lres : d_el_lres);

AR temp,suiv : d_el_lres;

EGIN
  temp := lres;
  WHILE temp <> NIL DO
     BEGIN
        suiv := temp^.next;
        DISPOSE(temp);
        temp := suiv;
     END;
  lres := NIL;
ND;

==========================================================================}
```

```
   fonction X_conv :
     IN : fich : fichier de X
     valeur de la fonction : designateur de la liste de X
     semantique : convertit le fichier "fich" de 'X' en une liste de
                  'X' et retourne un designateur vers la tete de cette liste.


 Ordre de la liste lineaire respecte ordre du fichier }

UNCTION lnom_conv(VAR fich : f_el_nom) : d_el_lnom;

AR liste_nom,endroit_ins : d_el_lnom;

EGIN
  RESET(fich);
  liste_nom := lnom_cree;
  endroit_ins := VIDE;
  WHILE NOT EOF(fich) DO
     BEGIN
        lnom_ins(liste_nom,fich^,endroit_ins);
        endroit_ins := lnom_suivant(liste_nom,endroit_ins);
        GET(fich);
     END;
  lnom_conv := liste_nom;
ND;

----------------------------------------------------------------------------}

UNCTION ldate_conv(VAR fich : f_el_date) : d_el_ldate;

AR liste_date,endroit_ins : d_el_ldate;

EGIN
  RESET(fich);
  liste_date := ldate_cree;
  endroit_ins := VIDE;
  WHILE NOT EOF(fich) DO
     BEGIN
        ldate_ins(liste_date,fich^,endroit_ins);
        endroit_ins := ldate_suivant(liste_date,endroit_ins);
        GET(fich);
     END;
  ldate_conv := liste_date;
ND;

----------------------------------------------------------------------------}

UNCTION lmc_conv(VAR fich : f_el_mc) : d_el_lmc;

AR liste_mc,endroit_ins : d_el_lmc;

EGIN
  RESET(fich);
  liste_mc := lmc_cree;
  endroit_ins := VIDE;
  WHILE NOT EOF(fich) DO
     BEGIN
```

```
   fonction X_conv :
     IN : fich : fichier de X
     valeur de la fonction : designateur de la liste de X
     semantique : convertit le fichier "fich" de 'X' en une liste de
                  'X' et retourne un designateur vers la tete de cette liste.


 Ordre de la liste lineaire respecte ordre du fichier }

UNCTION lnom_conv(VAR fich : f_el_nom) : d_el_lnom;

AR liste_nom,endroit_ins : d_el_lnom;

EGIN
  RESET(fich);
  liste_nom := lnom_cree;
  endroit_ins := VIDE;
  WHILE NOT EOF(fich) DO
     BEGIN
        lnom_ins(liste_nom,fich^,endroit_ins);
        endroit_ins := lnom_suivant(liste_nom,endroit_ins);
        GET(fich);
     END;
  lnom_conv := liste_nom;
ND;

----------------------------------------------------------------------------}

UNCTION ldate_conv(VAR fich : f_el_date) : d_el_ldate;

AR liste_date,endroit_ins : d_el_ldate;

EGIN
  RESET(fich);
  liste_date := ldate_cree;
  endroit_ins := VIDE;
  WHILE NOT EOF(fich) DO
     BEGIN
        ldate_ins(liste_date,fich^,endroit_ins);
        endroit_ins := ldate_suivant(liste_date,endroit_ins);
        GET(fich);
     END;
  ldate_conv := liste_date;
ND;

----------------------------------------------------------------------------}

UNCTION lmc_conv(VAR fich : f_el_mc) : d_el_lmc;

AR liste_mc,endroit_ins : d_el_lmc;

EGIN
  RESET(fich);
  liste_mc := lmc_cree;
  endroit_ins := VIDE;
  WHILE NOT EOF(fich) DO
     BEGIN
```

```
        lmc_ins(liste_mc,fich^,endroit_ins);
        endroit_ins := lmc_suivant(liste_mc,endroit_ins);
        GET(fich);
      END;
  lmc_conv := liste_mc;
ND;

==========================================================================}
```

```
 procedures de traitement des dates }

ROCEDURE conv_date(VAR date_in : PACKED ARRAY[m..n:INTEGER] OF CHAR;
                   VAR date_out : str10;VAR code_ret : typ_code_ret);EXTERN;

 IN : date_in : date sous forme libre jour-mois-annee, en anglais ou en
                francais .
 OUT : date_out : date sous forme interne AAAA/MM/JJ.
 semantique : si la date introduite est valide, elle est transformee
              en date sous forme interne et le code retour vaut 'ok';
              sinon, le code retour indique l'erreur qui a ete decouverte

     valeur possibles pour le code retour :
         join : jour doit etre inferieur au jour courant
         moin : mois doit etre inferieur au mois courant
         anin : annee doit etre inferieure a l'annee courante
         jnum : jour doit etre numerique
         lj2 : longueur du champ jour doit etre inferieure a 2
         lm2 : longueur du champ mois (numerique) doit etre inferieure a 2
         lm10 : longueur du champ mois (alpha) doit etre inferieure a 2
         mnc : mois non connu
         anum : annee doit etre numerique
         rmo : range mois incorrect (doit etre entre 1 et 12)
         rj28
         rj29
         rj30
         rj31 : range jour incorrect


-------------------------------------------------------------------------}

ROCEDURE pretty_date(date_in : str10 ;VAR  jolie_date : str20);EXTERN;

 IN : date_in : date sous forme interne
 OUT : date_out : date sous une forme agreable a lire (jour, mois en toutes
                  lettres et annee en quatre chiffres).
 semantique : convertit la date (supposee valide) en une date agreable
              a lire pour un etre humain.


=========================================================================}
```

```
ROCEDURE lnom_search(deb_rech : d_el_lnom;nom_cherche : str20;
                     VAR d_nom_trouve : d_el_lnom);

 IN : deb_rech : designateur d'un element dans une liste de noms
      nom_cherche : contenu d'un element de la liste des noms
 OUT : d_nom_trouve : designateur d'un element de la liste des noms
 semantique : cherche dans une liste de noms le nom "nom_cherche",
              a partir de l'endroit designe par "deb_rech".
              "d_nom_trouve" designe l'element correspondant
              s'il existe, ou vaut VIDE s'il n'existe pas.


AR courant : d_el_lnom;
   nom_cour,l_nom_cherche,l_nom_cour : str20;

EGIN
  IF deb_rech = VIDE THEN d_nom_trouve := VIDE
     ELSE
        BEGIN
           lower(nom_cherche,l_nom_cherche);
           courant := deb_rech;
           lnom_content(courant,nom_cour);
           lower(nom_cour,l_nom_cour);
           WHILE (l_nom_cherche <> l_nom_cour) AND (courant <> VIDE) DO
              BEGIN
                 courant := lnom_suivant(deb_rech,courant);
                 IF courant <> VIDE THEN
                    BEGIN
                       lnom_content(courant,nom_cour);
                       lower(nom_cour,l_nom_cour);
                    END;
              END;
           d_nom_trouve := courant;
        END;
ND;

----------------------------------------------------------------------}

ROCEDURE ldate_sup_search(deb_rech : d_el_ldate;date_cherche : str10;
                          VAR d_date_trouve : d_el_ldate);

 IN : deb_rech : designateur d'un element dans une liste de dates
      date_cherche : contenu d'un element d'une liste des dates (date sous
                     forme interne)
 OUT : d_date_trouve : designateur d'un element d'une liste des dates
 semantique : cherche dans une liste de dates la premiere date posterieure
              a  "date_cherche", a partir de l'endroit designe par
              "debut_rech". "d_date_trouve" designe l'element
              correspondant s'il existe, ou vaut VIDE s'il n'existe pas.


AR courant : d_el_ldate;
   date_cour : str10;

EGIN
  courant := VIDE;
  IF deb_rech = VIDE THEN d_date_trouve := VIDE
```

```
    ELSE
       BEGIN
          courant := ldate_suivant(deb_rech,VIDE);
          ldate_content(courant,date_cour);
          WHILE date_cherche >= date_cour DO
             BEGIN
                courant := ldate_suivant(deb_rech,courant);
                IF courant = VIDE THEN date_cour := maxstr10
                                  ELSE ldate_content(courant,date_cour);
             END;
          d_date_trouve := courant;
       END;
ND;

----------------------------------------------------------------------------}

ROCEDURE lres_search(deb_rech : d_el_lres;res_cherche : INTEGER;
                     VAR d_res_trouve : d_el_lres);

 IN : deb_rech : designateur d'un element d'une liste de resultats
      res_cherche : contenu d'un element d'une liste de resultats ( cad
                    "adresse" dans la collection des fiches signaletiques)
 OUT : d_res_trouve : designateur d'un element dans une liste de resultats
 semantique : cherche dans une liste de resultats le resultat "res_cherche",
              a partir de l'endroit designe par "deb_rech". "d_res_trouve"
               designe l'element correspondant s'il existe, ou vaut VIDE
               s'il n'existe pas.


AR courant : d_el_lres;
   res_cour : INTEGER;

EGIN
  IF deb_rech = VIDE THEN d_res_trouve := VIDE
     ELSE
        BEGIN
           courant := lres_suivant(deb_rech,VIDE);
           lres_content(courant,res_cour);
           WHILE res_cherche <> res_cour DO
              BEGIN
                 courant := lres_suivant(deb_rech,courant);
                 IF courant = VIDE THEN res_cour := res_cherche
                                   ELSE lres_content(courant,res_cour);
              END;
           d_res_trouve := courant;
        END;
ND;

----------------------------------------------------------------------------}

ROCEDURE lmc_simili_search(deb_rech : d_el_lmc;mc_cherche : str20;
                     VAR d_mc_trouve : d_el_lmc;VAR delta : INTEGER);

 IN : deb_rech : designateur d'un element d'une liste de mots-cles
      mc_cherche : contenu d'un element d'une liste de mots-cles
 OUT : d_mc_trouve : designateur d'un element d'une liste de mots-cles
 semantique : cherche dans une liste de mots-cles le premier mot-cle
              similaire a "mc_cherche", a partir de l'endroit designe
              par "deb_rech". "d_mc_trouve" designe l'element
              correspondant s'il existe, ou vaut VIDE s'il n'existe pas.
```

```
                "delta" vaut  0 si le mot trouve est egal au mot cherche
                       est < 0 si on n'a pas trouve de mot similaire
                       indique le nombre de differences si le mot trouve
                                 est similaire au mot cherche.


AR courant : d_el_lmc;
   mc_cour,l_mc_cherche,l_mc_cour : str20;

EGIN
  IF deb_rech = VIDE THEN BEGIN delta := -1;d_mc_trouve := VIDE END
     ELSE
        BEGIN
           lower(mc_cherche,l_mc_cherche);
           courant := deb_rech;
           lmc_content(courant,mc_cour);
           lower(mc_cour,l_mc_cour);
           delta := simili(l_mc_cherche,l_mc_cour);
           WHILE (delta < 0) AND (courant <> VIDE) DO
              BEGIN
                 courant := lmc_suivant(deb_rech,courant);
                 IF courant <> VIDE THEN
                    BEGIN
                       lmc_content(courant,mc_cour);
                       lower(mc_cour,l_mc_cour);
                       delta := simili(l_mc_cherche,l_mc_cour);
                    END;
              END;
           d_mc_trouve := courant;
        END;
ND;

==========================================================================}

ROCEDURE dacces(VAR fich : TEXT;debut : INTEGER);EXTERN;

 IN : fich : fichier de caracteres
      debut : numero de la ligne de l'ecran a partir de laquelle on desire
              que l'affichage se passe
      commandes de l'utilisateur au terminal
 OUT : affichage du contenu du fichier
 semantique : affiche le contenu du fichier "fich" sur l'ecran d'un
              terminal, a partir de la ligne "debut"
              L'utilisateur peut donner les commandes suivantes :
                   <space> pour avancer d'un ecran dans le fichier
                   <backspace> pour reculer d'un ecran
                   'q' pour terminer l'affichage
                   control-p pour imprimer le contenu du fichier


--------------------------------------------------------------------------}

ROCEDURE affich(VAR fich : TEXT);

 cf DACCES; le parametre "debut" etant fixe a la CONSTANTE "sepa",
 definie au debut du fichier }

EGIN
  dacces(fich,sepa)
ND;
```

```
ROCEDURE lres_aff(lres : d_el_lres);
 IN : lres : designateur d'une liste de resultats ( cad d'"adresses" de
             fiches signaletiques)
 OUT : affichage au terminal
 parametre global : fichier des fiches signaletiques "fhlp", ouvert en
                    lecture. La position courante de ce fichier est
                    modifiee.
 semantique : affiche au moyen de la routine "affich" les noms et fonctions
              des programmes correspondants aux elements du fichier d'aide
              "fhlp" dont les adresses sont contenues dans la liste
              designee par "lres" }

AR cour : d_el_lres;
   ftemp : TEXT;
   contenu_cour : INTEGER;

EGIN
  RESET(ftemp);
  cour := lres;
  WRITELN(ftemp,'Voici les renseignements sur les programmes correspondants aux criteres : ');
  WRITELN(ftemp);
  WHILE cour <> VIDE DO
     BEGIN
        lres_content(cour,contenu_cour);
        setpos(fhlp,contenu_cour);
        WRITELN(ftemp,'Nom du programme : ',fhlp^.nom);
        WRITELN(ftemp,'Fonction : ');
        WRITELN(ftemp,fhlp^.fct);
        WRITELN(ftemp);
        cour := lres_suivant(lres,cour);
     END;
  affich(ftemp);
  delete(ftemp);
ND;

----------------------------------------------------------------------------}

ROCEDURE lmc_aff(lmc : d_el_lmc);

 IN : lmc : designateur d'une liste de mots-cles
 OUT : affichage
 semantique : affiche sur trois colonnes, au moyen de la routine "affich",
              les mots-cles contenus dans la liste designee par "lmc"


AR cour,next : d_el_lmc;
   ftemp : TEXT;
   i : INTEGER;
   contenu_cour,contenu_next,l_contenu_cour,l_contenu_next : str20;


EGIN
  i := 1;
  RESET(ftemp);
  WRITELN(ftemp,'Voici une liste des mots-cles existants : ');
  WRITELN(ftemp);

  cour := lmc_suivant(lmc,vide);
```

```
WHILE code_ret = inter DO
   BEGIN
      RESET(fhgen,nom_meta_fich_aide);
      dacces(fhgen,1);
      touch(w_menu_gen);
      refresh(w_menu_gen);
      readstr(21,25,longueur,str,count,[CHR(13)],normal,code_ret);
      wipefield(w_menu_gen,count);
      wipe(w_menu_gen);
      refresh(w_menu_gen);
   END;
strset(fstr,str);
READ(fstr,typ_com:count:break_set);
lower(typ_com,l_typ_com);
discard(break_set,fstr);
READ(fstr,suite);
IF match(l_typ_com,'mots-cles') THEN
   BEGIN
      com_cour.intitule := 'M';
      IF suite = ligne_blanche THEN
         BEGIN
            ldem_intro(suite,code_ret);
            IF code_ret = callguru THEN
               BEGIN
                  aff_msg(callguru);
                  GOTO 100;
               END;
         END;
      lmc_decode(suite,com_cour,code_ret);
      WHILE (code_ret <> ok) AND (code_ret <> termin) DO
         BEGIN
            aff_msg(code_ret);
            ldem_intro(suite,code_ret);
            IF code_ret = callguru THEN
               BEGIN
                  aff_msg(callguru);
                  GOTO 100;
               END;
            lmc_decode(suite,com_cour,code_ret);
         END;
   END

ELSE IF match(l_typ_com,'date') THEN
   BEGIN
      com_cour.intitule := 'D';
      IF suite = ligne_blanche THEN
         BEGIN
            date_intro(suite,code_ret);
            IF code_ret = callguru THEN
               BEGIN
                  aff_msg(callguru);
                  GOTO 100;
               END;
         END;
      date_decode(suite,com_cour,code_ret);
      WHILE (code_ret <> ok) AND (code_ret <> termin) DO
         BEGIN
            aff_msg(code_ret);
            date_intro(suite,code_ret);
            IF code_ret = callguru THEN
```

```
PROCEDURE intro_complet(VAR com_cour : typ_commande;
                        VAR code_ret : typ_code_ret);

{ IN : introduction au terminal
  OUT : com_cour : commande sous forme interne
        code_ret : code retour
  parametres globaux : fenetre "w_menu_gen", definie, garnie ou pas
                       fichier des fenetres de nom "nom_fich_fenetre"
                       fichier-texte d'aide du programme AIDE, de nom
                            "nom_meta_fich_aide"
                       ligne_blanche : chaine de caracteres, de longueur
                                       egale a la largeur maximale de
                                       l'ecran, garnie de blancs
  Semantique : A partir du terminal et dans le cadre de l'ecran "w_menu_gen",
               garnit la partie "intitule" de la commande "com_cour" ainsi
               que eventuellement,la 'deuxieme partie correspondante.
                     ( nom d'un programme, date de recherche ou liste de
                       mots-cles cherches)
               En cas d'introduction de "?" comme premier caractere, le texte
               du fichier nomme "nom_meta_fich_aide" est affiche en guise
               d'aide.

  Code_retour : ok, callguru }

LABEL 100;

VAR fstr : TEXT;
    str : PACKED ARRAY [1..lim_col] OF CHAR;
    longueur,count : INTEGER;
    typ_com,l_typ_com : str9;
    suite : PACKED ARRAY [1..lim_col] OF CHAR;
    fmgen : TEXT;
    break_set : char_set;

BEGIN

   IF w_menu_gen = vide THEN
      BEGIN
         retrieve(w_menu_gen,'menu-gen  ',nom_fich_fenetre,code_ret);
         IF code_ret <> ok THEN
            BEGIN
               code_ret := callguru;
               aff_msg(code_ret);
               GOTO 100;
            END;
      END;

   break_set := [' ',',','.',';',':'];
   suite := ligne_blanche;
   longueur := maxy(w_menu_gen) - 25;

   REPEAT
   touch(w_menu_gen);
   refresh(w_menu_gen);
   move(w_menu_gen,21,25);
   readstr(21,25,longueur,str,count,[CHR(13)],normal,code_ret);
   wipefield(w_menu_gen,count);
   wipe(w_menu_gen);
   refresh(w_menu_gen);
```

```
            refresh(w_aide_date);
            aff_msg(mic);
            wipe(w_aide_date);
            refresh(w_aide_date);
            touch(w_dem_date);
            refresh(w_dem_date);
            readstr(1,63,longueur,str,i,[CHR(13)],normal,code_ret);
         END;
      wipefield(w_dem_date,count);
      wipe(w_dem_date);
      refresh(w_dem_date);
100:
END;

{=============================================================================}
```

```
100:
END;

{-------------------------------------------------------------------------------}

PROCEDURE date_intro(VAR str : PACKED ARRAY [m..n:INTEGER ] OF CHAR ;
                     VAR code_ret : typ_code_ret);

{ IN : introduction au terminal
  OUT : str : chaine de caractere
        code_ret : code retour
  parametres globaux : fenetre "w_dem_date" et "w_aide_date" definies,
                       garnies ou pas
                     fichier des fenetres de nom "nom_fich_fenetre"
  semantique : garnit la chaine "str" avec
               date lue au terminal dans le cadre de l'ecran "w_dem_date".
               En cas d'introduction de "?" comme
               premier caractere, affiche la fenetre "w_aide_date" comme
               aide.

  Valeurs possibles pour le code retour :
    ok
    calguru : erreur imprevue necessitant l'appel d'une personne competente
}

LABEL 100;

VAR longueur,count : INTEGER;

BEGIN
   IF w_dem_date = vide THEN
      BEGIN
         retrieve(w_dem_date,'dem_date  ',nom_fich_fenetre,code_ret);
         IF code_ret <> ok THEN
            BEGIN
               code_ret := calguru;
               aff_msg(code_ret);
               GOTO 100;
            END;
      END;

   IF w_aide_date = vide THEN
      BEGIN
         retrieve(w_aide_date,'aide_date ',nom_fich_fenetre,code_ret);
         IF code_ret <> ok THEN
            BEGIN
               code_ret := calguru;
               aff_msg(code_ret);
               GOTO 100;
            END;
      END;

   touch(w_dem_date);
   refresh(w_dem_date);
   move(w_dem_date,1,63);
   longueur := maxy(w_dem_date) - 63;
   readstr(1,63,longueur,str,i,[CHR(13)],normal,code_ret);
   WHILE code_ret = inter DO
      BEGIN
         touch(w_aide_date);
```

```
  parametres globaux : fenetre "w_dem_mc", definie, garnie ou pas
                       fichier des fenetres de nom "nom_fich_fenetre"
                       fenetre "w_aff_dem", definie, existante ou pas
                       fichier auxiliaire des mots_cles, de nom interne "fmc"
                                  ouvert en lecture.

                       fichier des fiches signaletiques

  semantique : garnit la chaîne "str" avec la
               liste de mots_cles introduits du terminal dans le cadre de
               l'ecran "w_dem_mc".
               affiche aussi le contenu de la fenetre "w_aff_dem". (Cette
               fenetre est garnie par la routine "rech_et_aff_mc" avec
               les mots-cles deja introduits).
               En cas d'introduction de "?" comme premier caractere, affiche
               en guise d'aide, la liste des mots-cles existants .

   Valeurs possibles pour le code retour :
    ok
    callguru : erreur imprevue necessitant l'appel d'une personne competente
}

LABEL 100;

VAR longueur,count : INTEGER;

BEGIN
   IF w_dem_mc = vide THEN
      BEGIN
         retrieve(w_dem_mc,'dem_mc     ',nom_fich_fenetre,code_ret);
         IF code_ret <> ok THEN
            BEGIN
               code_ret := callguru;
               aff_msg(code_ret);
               GOTO 100;
            END;
      END;
IF w_aff_dem = vide THEN  w_aff_dem := newwin(2,1,2,80);


touch(w_dem_mc);
refresh(w_dem_mc);
touch(w_aff_dem);
refresh(w_aff_dem);
move(w_dem_mc,1,60);
longueur := maxy(w_aff_dem) - 60;
readstr(1,60,longueur,str,count,[CHR(13)],normal,code_ret);
WHILE code_ret = inter DO
   BEGIN
      IF lmc = VIDE THEN lmc := lmc_conv(fmc);
      lmc_aff(lmc);
      touch(w_dem_mc);
      refresh(w_dem_mc);
      touch(w_aff_dem);
      refresh(w_aff_dem);
      readstr(1,60,longueur,str,count,[CHR(13)],normal,code_ret);
   END;
wipefield(w_dem_mc,count);
wipe(w_dem_mc);
refresh(w_dem_mc);
```

```
{ IN : introduction au terminal
  OUT : str : chaine de caracteres
        code_ret : code retour
  parametres globaux : fenetre "w_dem_nom" existante, garnie ou valant VIDE
                       fichier des fenetres de nom "nom_fich_fenetre"
                       fichier auxiliaire des noms, de nom interne "fnom",
                            ouvert en lecture
  semantique : garnit le string  "str" avec le premier mot
               lu au terminal dans le cadre de l'ecran "w_dem_nom".
               En cas d'introduction de "?" comme premier caractere, affiche
               la liste des noms de programme existants comme aide.

   Valeurs possibles pour le code retour :
    ok
    callguru : erreur imprevue necessitant l'appel d'une personne competente
}

LABEL 100;

VAR longueur,count : INTEGER;

BEGIN
   IF w_dem_nom = vide THEN
      BEGIN
         retrieve(w_dem_nom,'dem_nom    ',nom_fich_fenetre,code_ret);
         IF code_ret <> ok THEN
            BEGIN
               code_ret := callguru;
               aff_msg(code_ret);
               GOTO 100;
            END;
      END;

   touch(w_dem_nom);
   refresh(w_dem_nom);
   move(w_dem_nom,1,45);
   longueur := maxy(w_dem_nom) - 45;
   readstr(1,45,longueur,str,count,[CHR(13)],normal,code_ret);
   WHILE code_ret = inter DO
      BEGIN
         IF lnom = VIDE THEN lnom := lnom_conv(fnom);
         lnom_aff(lnom);
         touch(w_dem_nom);
         refresh(w_dem_nom);
         readstr(1,45,longueur,str,count,[CHR(13)],normal,code_ret);
      END;
   wipefield(w_dem_nom,count);
   wipe(w_dem_nom);
   refresh(w_dem_nom);
100:
END;

{-------------------------------------------------------------------------}

PROCEDURE ldem_intro(VAR str : PACKED ARRAY [m..n:INTEGER] OF CHAR ;
                     VAR code_ret : typ_code_ret);

{ IN : introduction au terminal
  OUT : str : chaine de caracteres
        code_ret : code retour
```

```
   code_ret := ok;
   mot_cle := b120;
   l_mot_cle := b120;
   endroit_ins := VIDE;
   strset(fstr,str);
   READ(fstr,mot_cle:count:break_set);
   IF match(mot_cle,'quitter') OR (mot_cle = b120) THEN
      BEGIN
         code_ret := termin;
         mot_cle := b120;
      END;
   WHILE mot_cle <> b120 DO
      BEGIN
         lower(mot_cle,l_mot_cle);
         ldem_ins(com_cour.ldem,l_mot_cle,endroit_ins);
         endroit_ins := ldem_suivant(com_cour.ldem,endroit_ins);
         discard(break_set,fstr);
         mot_cle := b120;
         READ(fstr,mot_cle:count:break_set);
      END;
END;

{-------------------------------------------------------------------------}

PROCEDURE nom_decode(VAR str : PACKED ARRAY [m..n:INTEGER] OF CHAR;
                     VAR com_cour : typ_commande;VAR code_ret : typ_code_ret);

{ IN : str : chaine de caracteres
  OUT : com_cour : commande sous forme interne
        code_ret : code retour
  semantique : garnit la partie "nom_fich" de la commande "com_cour" avec
               le nom qui se trouve dans "str"; le code-retour vaut alors
               'ok'. Si "str" correspond a 'quitter' ou est vide,le
               code-retour vaut 'termin'.

        Valeurs possibles pour le code retour :
               ok, termin
}

VAR i : INTEGER;
    fstr: TEXT;
    nom : str20;
    break_set : char_set;

BEGIN
   com_cour.nom_fich := b120;
   break_set := [' ',',',';',':'];
   code_ret := ok;
   strset(fstr,str);
   READ(fstr,nom:i:break_set);
   IF match(nom,'quitter') OR (nom = b120)
            THEN code_ret := termin
            ELSE com_cour.nom_fich := nom;
END;

{-------------------------------------------------------------------------}

PROCEDURE nom_intro(VAR str : PACKED ARRAY [m..n:INTEGER] OF CHAR ;
                    VAR code_ret : typ_code_ret);
```

```
PROCEDURE date_decode(VAR str : PACKED ARRAY [m..n:INTEGER] OF CHAR;
                      VAR com_cour : typ_commande;VAR code_ret :typ_code_ret);

{ IN : str : chaine de caracteres
  OUT : com_cour : commande en format interne
        code_ret : code retour
  semantique :essaie de convertir (au moyen de la routine "conv_date") le
              contenu du string "str" en une date en format interne
              (AAAA/MM/JJ), sauf si le string correspond a "quitter" ou
              est vide,dans lequel cas on retourne un code-retour
              valant 'termin'.
              Si la conversion reussit, la date convertie se trouve dans
              la partie 'date_in' de la commande "com_cour" et le
              code_retour vaut 'ok';
              sinon, le code-retour indique l'erreur detectee

    Valeurs possibles pour le code retour :
       ok
       termin
       Une des valeurs resultantes de la conversion de date

}

BEGIN
   com_cour.date_in := bl10;
   code_ret := ok;
   IF match(str,'quitter') OR match(str,bl20)
                  THEN code_ret := termin
                  ELSE conv_date(str,com_cour.date_in,code_ret);
END;

{-------------------------------------------------------------------------}

PROCEDURE lmc_decode(VAR str : PACKED ARRAY [m..n:INTEGER] OF CHAR;
                     VAR com_cour : typ_commande;VAR code_ret : typ_code_ret);

{ IN : str : chaine de caracteres
  OUT : com_cour : commande sous forme interne
        code_ret : code retour
  semantique : convertit le string "str" en une liste de mots-cles qui
               sera designee par la partie "ldem" de la commande "com_cour";
               le code-retour "code_ret" vaut 'ok', sauf si le string
               correspond a "quitter" ou est vide, dans lequel cas il vaut
               'termin'

        Valeurs possibles du code retour :
            ok, termin
}

VAR fstr : TEXT;
    count : INTEGER;
    mot_cle,l_mot_cle : str20;
    endroit_ins : d_el_ldem;
    break_set : char_set;

BEGIN
   com_cour.ldem := VIDE;
   break_set := [',',';',':'];
```

```
move(w_aff_nom,7,7);
addstr(w_aff_nom,fhlp^.nom,normal);

move(w_aff_nom,10,1);
addstr(w_aff_nom,fhlp^.fct,normal);

pretty_date(fhlp^.date_maj,jolie_date);
move(w_aff_nom,12,8);
addstr(w_aff_nom,jolie_date,normal);

i := 1;
WHILE (i <= 3) DO
   BEGIN
      IF (fhlp^.docu[i] = ligne_blanche)
         THEN i := 4
            ELSE BEGIN
               move(w_aff_nom,14 + i,1);
               addstr(w_aff_nom,fhlp^.docu[i],normal);
               i := i + 1 ;
            END;
   END;
i := 1 ; j := 1;
strwrite(fstr,str);
WHILE (i <= nb_max_mc ) DO
   BEGIN
      nb_car_total := 0;
      str := ligne_blanche;
      WHILE (nb_car_total < (max_col-20)) AND (i <= nb_max_mc) DO
         BEGIN
            IF fhlp^.mc[i] <> bl20 THEN
               BEGIN
                  wstr(fstr,fhlp^.mc[i],count);
                  i := i + 1;
                  IF i <= nb_max_mc THEN IF fhlp^.mc[i] <> bl20 THEN WRITE(fstr,',');
                  nb_car_total := nb_car_total + count + 1;
               END
            ELSE i := nb_max_mc + 1; { pour terminer les DEUX boucles}

         END;

      move(w_aff_nom,19 + j,1);
      addstr(w_aff_nom,str,normal);

      j := j + 1;
      strwrite(fstr,str);
   END;
   END;
overlay(w_sauel_nom,w_aff_nom);
refresh(w_aff_nom);
aff_msg(mic);
clearall(w_aff_nom);
refresh(w_aff_nom);
.00:
:ND;

{=============================================================================}
```

```
    END;
  affich(ftemp);
  delete(ftemp);
END;

{-------------------------------------------------------------------------}

PROCEDURE el_hlp_aff(adresse : INTEGER);

{ IN : adresse : "adresse" d'une fiche signaletique du fichier d'aide "fhlp"
  OUT : affichage
  parametres globaux : fichier des fiches signaletiques "fhlp", ouvert en
                       lecture. La position courante de ce fichier est
                       modifiee.
                       fenetre "w_squel_nom", creees, garnies
                       ou valant VIDE
                       fenetre "w_aff_nom", creee ou valant VIDE
                       fichier des fenetres de nom "nom_fich_fenetre"
                       ligne_blanche : chaine de caracteres, de longueur
                                       egale a la largeur maximale de
                                       l'ecran, garnie de blancs

  semantique : affiche, en garnissant la fenetre "w_aff_nom" et en utilisant
               la fenetre "w_squel_nom", le contenu de l'element du fichier
                                         "fhlp" d'adresse "adresse",
                                          si cette adresse est >= 0
                                      le message "pas de renseignements"
                                          si cette adresse est < 0
}

LABEL 100;

VAR jolie_date : str20;
    i,j,count,nb_car_total,max_col : INTEGER;
    fstr : TEXT;
    str : PACKED ARRAY [1..lim_col] OF CHAR;

BEGIN
   IF w_squel_nom = vide THEN
      BEGIN
         retrieve(w_squel_nom,'squel_nom ',nom_fich_fenetre,code_ret);
         IF code_ret <> ok THEN
            BEGIN
               aff_msg(callguru);
               GOTO 100;
            END;
      END;
   IF w_aff_nom = vide THEN w_aff_nom := newwin(sepa,1,xphysx,xphysy);

   max_col := maxy(w_aff_nom);

   init_fields(w_aff_nom,FALSE);
   IF adresse < 0 THEN
      BEGIN
         move(w_aff_nom,7,7);
         addstr(w_aff_nom,'Pas de renseignements de base',reverse);
      END
   ELSE
      BEGIN
         setpos(fhlp,adresse);
```

```
  WHILE cour <> VIDE DO
     BEGIN
        next := lmc_suivant(lmc,cour);

        lmc_content(cour,contenu_cour);
        lower(contenu_cour,l_contenu_cour);

        lmc_content(next,contenu_next);
        lower(contenu_next,l_contenu_next);

        WHILE l_contenu_cour = l_contenu_next DO
           BEGIN
              next := lmc_suivant(lmc,next);
              lmc_content(next,contenu_next);
              lower(contenu_next,l_contenu_next);
           END;

        IF (i MOD 3) <> 0 THEN WRITE(ftemp,l_contenu_cour,bl10)
                          ELSE WRITELN(ftemp,l_contenu_cour);
        i := i + 1;

        cour := next;

     END;
  affich(ftemp);
  delete(ftemp);
END;

{-------------------------------------------------------------------------------}

PROCEDURE lnom_aff(lnom : d_el_lnom);

  IN : lnom : designateur d'une liste de noms
  OUT : affichage
  semantique : affiche sur trois colonnes, au moyen de la routine "affich",
               les noms contenus dans la liste designee par "lnom" }

VAR cour : d_el_lnom;
    ftemp : TEXT;
    i : INTEGER;
    contenu_cour : str20;

BEGIN
   cour := lnom_suivant(lnom,vide);
   i := 1;

   RESET(ftemp);
   WRITELN(ftemp,'Voici une liste des programmes au sujet desquels une aide de base existe : ');
   WRITELN(ftemp);

   WHILE cour <> VIDE DO
      BEGIN

         lnom_content(cour,contenu_cour);

         IF (i MOD 3) <> 0 THEN WRITE(ftemp,contenu_cour,bl10)
                           ELSE WRITELN(ftemp,contenu_cour);
         i := i + 1;
         cour := lnom_suivant(lnom,cour);
```

```
                BEGIN
                   aff_msg(callguru);
                   GOTO 100;
                END;
             date_decode(suite,com_cour,code_ret);
          END;
    END

ELSE IF match(l_typ_com,'nom') THEN
    BEGIN
       com_cour.intitule := 'N';
       IF suite = ligne_blanche THEN
          BEGIN
             nom_intro(suite,code_ret);
             IF code_ret = callguru THEN
                BEGIN
                   aff_msg(callguru);
                   GOTO 100;
                END;
          END;
       nom_decode(suite,com_cour,code_ret);
       WHILE (code_ret <> ok) AND (code_ret <> termin) DO
          BEGIN
             aff_msg(code_ret);
             nom_intro(suite,code_ret);
             IF code_ret = callguru THEN
                BEGIN
                   aff_msg(callguru);
                   GOTO 100;
                END;
             nom_decode(suite,com_cour,code_ret);
          END;
       END

    ELSE IF match(l_typ_com,'quitter') THEN com_cour.intitule := 'Q'

    ELSE IF l_typ_com = '          ' THEN BEGIN
                                             code_ret := comvid;
                                             aff_msg(code_ret);
                                          END

    ELSE BEGIN
           code_ret := cin;
           aff_msg(code_ret);
         END;
UNTIL (code_ret = ok) OR (code_ret = callguru);
100:
END;

{=============================================================================}
```

```
PROCEDURE sel_mc(ldem : d_el_ldem;lmc : d_el_lmc;VAR lres : d_el_lres);

{ IN : ldem : designateur d'une liste de noms de programme
       lmc : designateur d'une liste de mots-cles
       lres : designateur d'une liste de resultats ( cad d'adresses de
              fiches signaletiques )
  OUT :lres : designateur d'une nouvelle liste de resultats ( cad
              d'adresses de fiches signaletiques )
  parametre global : fenetre "w_mc_sim" definie, creee ou pas
  semantique : si "lres" vaut VIDE lors de l'appel :
                        selectionne dans la liste designee par "lmc",
                        les elements dont la  partie "contenu" est "similaire"
                        ou egale aux mots se trouvant dans la liste designee
                        par "ldem".
                        Ces elements sont mis dans une liste que designera
                        "lres" apres l'appel.
               si "lres" pointe deja vers une liste :
                        la selection s'effectue sur un double critere :
                           appartenance a "lmc"  ET   appartenance a "lres"
               On emploie la fenetre "w_mc_sim" pour traiter les cas de
               "similarite"
}

VAR dem_cour : d_el_ldem;
    contenu_dem_cour : str20;
    anc_lres : d_el_lres;
    el_mc_trouve,temp_el_mc : d_el_lmc;
    contenu_el_mc_trouve,contenu_temp_el_mc : str20;
    delta : INTEGER;
    el_res_trouve : d_el_lres;
    contenu_el_res_trouve : INTEGER;
    bidon : d_el_lres;
    endroit_de_rech : d_el_lmc;
    insertion : BOOLEAN;
    adresse_el_mc_trouve : INTEGER;

BEGIN
   IF w_mc_sim = vide THEN w_mc_sim := newwin(seba,1,xphysx,xphysy);

   bidon := VIDE;
   dem_cour := ldem_suivant(ldem,VIDE);  {cad PREMIER}
   WHILE dem_cour <> VIDE DO
      BEGIN
         anc_lres := lres;
         lres := VIDE;
         endroit_de_rech := lmc_suivant(lmc,VIDE);    {cad PREMIER}
         REPEAT
            insertion := FALSE;
            ldem_content(dem_cour,contenu_dem_cour);
            lmc_simili_search(endroit_de_rech,contenu_dem_cour,el_mc_trouve,delta);
            IF delta = 0 THEN
               BEGIN
                  endroit_de_rech := lmc_suivant(lmc,el_mc_trouve);
                  insertion := TRUE;
               END;
            IF delta > 0 THEN
               BEGIN
                  lmc_content(el_mc_trouve,contenu_el_mc_trouve);
```

```
endroit_de_rech := lmc_suivant(lmc,el_mc_trouve);

move(w_mc_sim,6,1);
addstr(w_mc_sim,'Pas de mot-cle strictement correspondant.',normal);
move(w_mc_sim,7,1);
addstr(w_mc_sim,'Le mot-cle "',normal);
addstr(w_mc_sim,contenu_el_mc_trouve,normal);
addstr(w_mc_sim,'" est similaire au mot-cle "',normal);
addstr(w_mc_sim,contenu_dem_cour,normal);
addstr(w_mc_sim,'".',normal);
move(w_mc_sim,8,1);
addstr(w_mc_sim,'Convient-il ? ',normal);
refresh(w_mc_sim);
IF conf_msg(dco) THEN
   BEGIN
      set_ldem_content(dem_cour,contenu_el_mc_trouve);
                { le contenu du mc trouve -> demande cour}
      endroit_de_rech := lmc_suivant(lmc,el_mc_trouve);
      insertion := TRUE;
   END
ELSE
                { on passe au dessus de tous les mots-cles
                  egaux a celui qu'on vient de refuser,
                  pour reprendre la recherche apres ceux-ci}
   BEGIN
      temp_el_mc := lmc_suivant(lmc,el_mc_trouve);
      lmc_content(temp_el_mc,contenu_temp_el_mc);
      WHILE contenu_el_mc_trouve = contenu_temp_el_mc DO
         BEGIN
            temp_el_mc := lmc_suivant(lmc,temp_el_mc);
            lmc_content(temp_el_mc,contenu_temp_el_mc);
         END;
        endroit_de_rech := temp_el_mc;
      END;
      clearall(w_mc_sim);
      refresh(w_mc_sim);
   END;
IF insertion THEN
   BEGIN
      IF anc_lres <> VIDE THEN
         BEGIN
            lmc_adresse(el_mc_trouve,adresse_el_mc_trouve);
            lres_search(anc_lres,adresse_el_mc_trouve,el_res_trouve);
            IF el_res_trouve <> VIDE
               THEN BEGIN
                     lres_content(el_res_trouve,contenu_el_res_trouve);
                     lres_ins(lres,contenu_el_res_trouve,bidon);
                    END;
         END
      ELSE
         lres_ins(lres,el_mc_trouve^.adresse,bidon);
   END;
UNTIL delta < 0;

lres_discard(anc_lres);

dem_cour := ldem_suivant(ldem,dem_cour);

IF lres = VIDE THEN dem_cour := VIDE; { pour terminer la boucle }
```

```
      END;
END;

{-------------------------------------------------------------------------------}

PROCEDURE sel_date(debut : d_el_ldate;date_in : str10;VAR lres : d_el_lres);

{ IN : debut : designateur d'un endroit (cad d'un element) dans une liste
               de dates
       date_in : contenu de l'element d'une liste de dates
  OUT : lres :  designateur d'une liste de "resultats"
  semantique : selectionne dans la liste designee par  "debut",
               les dates posterieures a "date_in" (exprimee sous forme
               AAAA/MM/JJ), et les met dans une liste qui sera designee par
               "lres"
}

VAR date_trouvee : d_el_ldate;
    res_vide : d_el_lres;
    adresse_date_trouvee : INTEGER;

BEGIN
   res_vide := VIDE;
   lres := VIDE;
   ldate_sup_search(debut,date_in,date_trouvee);
   WHILE date_trouvee <> VIDE DO
      BEGIN
         ldate_adresse(date_trouvee,adresse_date_trouvee);
         lres_ins(lres,adresse_date_trouvee,res_vide);
         date_trouvee := ldate_suivant(ldate,date_trouvee);
      END;
END;

{===============================================================================}
```

```
{ Les trois procedures suivantes gerent le traitement d'une commande de type
  "aide sur un programme de nom donne" , "aide sur les programmes m-a-j a
  partir d'une date donnee", "aide sur les programmes correspondant a des
  mots-les donnes".
}

PROCEDURE rech_et_aff_nom(VAR cde : typ_commande);
{ IN : cde : commande sous forme interne;
  parametre global : lnom : designateur vers une liste de noms
                            fichier auxiliaire des noms, de nom interne "fnom",
                            ouvert en lecture.

                            fichier des fiches signaletiques

  semantique : "nom" doit designer la liste des noms des programmes dont on
               possede une fiche signaletique. (si ce n'est pas le cas,
               cette liste est creee a partir du fichier
               auxiliaire des noms.)
               Cette procedure affiche alors la fiche signaletique
               correspondant au programme dont le nom est dans la partie "nom"
               de "cde".
}

LABEL 100;

VAR d_nom_trouve : d_el_lnom;
    adresse_d_nom_trouve : INTEGER;
    nom_hlp : str24;
    nom_hlp_complet : str28;
    texte_hlp : TEXT;

BEGIN
   IF lnom = VIDE THEN lnom := lnom_conv(fnom);
   lnom_search(lnom,cde.nom_fich,d_nom_trouve);
   IF d_nom_trouve = VIDE THEN el_hlp_aff(-1)
                          ELSE BEGIN
                                 lnom_adresse(d_nom_trouve,adresse_d_nom_trouve);
                                 el_hlp_aff(adresse_d_nom_trouve);
                               END;
   concat(cde.nom_fich,'.hlp',nom_hlp);
   concat('hlp:',nom_hlp,nom_hlp_complet);
   RESET(texte_hlp,nom_hlp_complet,'/O');
   IF erstat(texte_hlp) = 0 THEN
                  IF conf_msg(chlp) THEN affich(texte_hlp);
   cde.nom_fich := b120;

100:
END;

{-------------------------------------------------------------------------------}

PROCEDURE rech_et_aff_date(VAR cde : typ_commande);
{ IN : cde : commande sous forme interne
  parametre global : ldate : designateur d'une liste de dates
                         fichier auxiliaire des dates, de nom interne "fdate",
                             ouvert en lecture.

                         fichier des fiches signaletiques.
```

```
  semantique : "ldate" doit designer la liste des dates de m-a-j des fiches
               signaletiques. (si ce n'est pas le cas, cette liste est creee
               a partir du fichier auxiliaire des dates.)
               Cette procedure cree alors la liste des adresses des fiches
               signaletiques mises a jour apres la partie "date" de "cde",
               et affiche les noms et descriptions des divers programmes.
}

VAR d_date_trouve : d_el_ldate;

BEGIN
   IF ldate = VIDE THEN ldate := ldate_conv(fdate);
   sel_date(ldate,cde.date_in,lres);
   lres_aff(lres);
   cde.date_in := b110;
END;

{-------------------------------------------------------------------------}

PROCEDURE rech_et_aff_mc(VAR cde : typ_commande);

{ IN : cde : commande sous forme interne
  parametre global : lmc : designateur d'une liste de mots-cles
                     fichier auxiliaire des mots-cles, de nom interne "fmc",
                         ouvert en lecture
                     w_aff_dem : designateur d'une fenetre, definie,
                                 creee ou pas.
  semantique : "lmc" doit designer la liste mots-cles exitants dans les
               diverses fiches signaletiques. (si ce n'est pas le cas,
               cette liste est creee a partir du fichier
               auxiliaire des mots-cles.)
               Cette procedure cree une liste des adresses des fiches
               signaletiques qui comprennent les mots-cles contenus
               dans la liste designee par la partie "ldem" de "cde". Les
               noms et descriptions de ces programmes sont affiches.
               Il est possible d'affiner la recherche en introduisant des
               mots-cles supplementaires.
               La fenetre "w_aff_dem" est initialisee si necessaire et
               garnie des mots_cles introduits.
}

VAR dem_cour : d_el_ldem;
    contenu_dem_cour : str20;
    fini : BOOLEAN;
    str : PACKED ARRAY [1..lim_col] OF CHAR;

BEGIN
   IF w_aff_dem = vide THEN w_aff_dem := newwin(2,1,2,80);

   move(w_aff_dem,2,1);
   addstr(w_aff_dem,'Mots-cles introduits : ',normal);
   fini := FALSE;
   IF lmc = VIDE THEN lmc := lmc_conv(fmc);
   lres_discard(lres);

   dem_cour := cde.ldem;
   WHILE dem_cour <> VIDE DO
      BEGIN
         ldem_content(dem_cour,contenu_dem_cour);
```

```
            addstr(w_aff_dem,contenu_dem_cour,normal);
            addstr(w_aff_dem,', ',normal);
            dem_cour := ldem_suivant(cde.ldem,dem_cour);
         END;
      WHILE (NOT fini) AND (code_ret <> termin) DO
         BEGIN
            sel_mc(cde.ldem,lmc,lres);
            IF lres = VIDE THEN
               BEGIN
                  aff_msg(ppc);
                  fini := TRUE;
               END
            ELSE
               BEGIN
                  lres_aff(lres);
                  ldem_discard(cde.ldem);
                  ldem_intro(str,code_ret);
                  lmc_decode(str,cde,code_ret);
                  dem_cour := cde.ldem;
                  WHILE dem_cour <> VIDE DO
                     BEGIN
                        ldem_content(dem_cour,contenu_dem_cour);
                        addstr(w_aff_dem,contenu_dem_cour,normal);
                        addstr(w_aff_dem,', ',normal);
                        dem_cour := ldem_suivant(cde.ldem,dem_cour);
                     END;
               END;
         END;
      ldem_discard(cde.ldem);
      erase(w_aff_dem);
   END;
```

```
{-------------------------------------------------------------------------}
```

```
{programme principal}

BEGIN

{initialisations}

   initscr(code_ret);
   IF code_ret <> ok THEN BEGIN ecr_msg(code_ret);GOTO 100 END;

   FOR i := 1 TO lim_col DO ligne_blanche[i] := ' ';

   RESET(fhlp,nom_fich_aide);
   RESET(fnom,nom_fich_nom);
   RESET(fmc,nom_fich_mc);
   RESET(fdate,nom_fich_date);
   lnom := VIDE;
   ldate := VIDE;
   lmc := VIDE;
   w_menu_gen := VIDE;
   w_dem_date := VIDE;
   w_dem_mc := VIDE;
   w_dem_nom := VIDE;
   w_aide_date := VIDE;
   w_sauel_nom := VIDE;
   w_mc_sim := VIDE;

{debut du programme }

   intro_complet(cde,code_ret);
   IF code_ret = callguru THEN
      BEGIN
         aff_msg(callguru);
         GOTO 99; {en cas de catastrophe}
      END;

   WHILE cde.intitule <> '0' DO
      BEGIN
         CASE cde.intitule OF
            'N' : rech_et_aff_nom(cde);
            'D' : rech_et_aff_date(cde);
            'M' : rech_et_aff_mc(cde);
         END;
         intro_complet(cde,code_ret);
         IF code_ret = callguru THEN
            BEGIN
               aff_msg(code_ret);
               GOTO 99;
            END;
      END;

99:endscr;
100:
   {branchement de fin en cas d'erreur d'initialisation de la gestion d'ecran}
END.
```

```
PROGRAM intro;

include 'sys:string.dec';
include 'sys:elems.dec';
include 'sys:cod_ret.dec';
include 'sys:utilit.dec';


VAR lnom : d_el_lnom;
    lmc : d_el_lmc;

    fnom : f_el_nom;
    fmc : f_el_mc;
    fdate : f_el_date;
    fhlp,ftemp : f_el_hlp;

    posit,pos_append : INTEGER;
    rep : str3;
    i : INTEGER;
    choix,l_choix : PACKED ARRAY [1..11] OF CHAR;

{ Pour memoire : fonctions en include


FUNCTION erstat (VAR f : FILE): INTEGER;EXTERN;
PROCEDURE getc(VAR c: CHAR);
FUNCTION confirm(c:CHAR):BOOLEAN;
PROCEDURE lower(VAR str,res : PACKED ARRAY [m..n:INTEGER] OF CHAR);
FUNCTION simili(VAR str1,str2 : PACKED ARRAY [m..n:INTEGER] OF CHAR):INTEGER;
}

{=============================================================================}

PROCEDURE conv_date(VAR date_in : PACKED ARRAY[m..n:INTEGER] OF CHAR;
                    VAR date_out : str10;VAR code_ret : typ_code_ret);EXTERN;

{-----------------------------------------------------------------------------}

PROCEDURE pretty_date(date_in : str10;VAR jolie_date :str20);EXTERN;

{=============================================================================}
```

```
FUNCTION lmc_cree : d_el_lmc;
BEGIN
   lmc_cree := NIL;
END;

FUNCTION lnom_cree : d_el_lnom;
BEGIN
   lnom_cree := NIL;
END;

{-------------------------------------------------------------------------}

FUNCTION lmc_suivant(liste : d_el_lmc;element : d_el_lmc) : d_el_lmc;
BEGIN
   IF element = NIL THEN lmc_suivant := liste
                    ELSE lmc_suivant := element^.next;
END;

FUNCTION lnom_suivant(liste : d_el_lnom;element : d_el_lnom) : d_el_lnom;
BEGIN
   IF element = NIL THEN lnom_suivant := liste
                    ELSE lnom_suivant := element^.next;
END;

{-------------------------------------------------------------------------}

PROCEDURE lmc_content(element : d_el_lmc;VAR contenu : str20);
BEGIN
   contenu := element^.content;
END;

PROCEDURE lnom_content(element : d_el_lnom;VAR contenu : str20);
BEGIN
   contenu := element^.content;
END;

{-------------------------------------------------------------------------}

PROCEDURE lnom_ins(VAR tete : d_el_lnom;contenu : el_nom;
                       VAR prec : d_el_lnom);

VAR nouv_nom : d_el_lnom;

BEGIN
   NEW(nouv_nom);
   nouv_nom^.content := contenu.content;
   nouv_nom^.adresse := contenu.adresse;
   IF prec = VIDE THEN
      BEGIN
         nouv_nom^.next := tete;
         tete := nouv_nom;
      END
   ELSE
      BEGIN
         nouv_nom^.next := prec^.next;
         prec^.next := nouv_nom;
      END;
END;
```

```
{-------------------------------------------------------------------------}

PROCEDURE lmc_ins(VAR tete : d_el_lmc; contenu : el_mc;
                        VAR prec : d_el_lmc);

VAR nouv_mc : d_el_lmc;

BEGIN
   NEW(nouv_mc);
   nouv_mc^.content := contenu.content;
   nouv_mc^.adresse := contenu.adresse;
   IF prec = VIDE THEN
      BEGIN
         nouv_mc^.next := tete;
         tete := nouv_mc;
      END
   ELSE
      BEGIN
         nouv_mc^.next := prec^.next;
         prec^.next := nouv_mc;
      END;
END;

{-------------------------------------------------------------------------}

PROCEDURE lmc_suppress(VAR liste:d_el_lmc;VAR el_a_sup : d_el_lmc);
{hypo : el_a_sup app liste}

VAR cour,prec : d_el_lmc;

BEGIN
   IF el_a_sup = liste THEN
      BEGIN
         liste := liste^.next;
         DISPOSE(el_a_sup);
      END
   ELSE
      BEGIN
         cour := liste;
         WHILE (cour <> el_a_sup ) AND (cour <> NIL) DO
            BEGIN
               prec := cour;
               cour := cour^.next;
            END;
         IF cour = NIL THEN WRITELN(TTY,'CATASTROPHE')
            ELSE
               BEGIN
                  prec^.next := cour^.next;
                  DISPOSE(el_a_sup);
               END;
      END;
END;

{-------------------------------------------------------------------------}

PROCEDURE lnom_suppress(VAR liste:d_el_lnom;VAR el_a_sup : d_el_lnom);
{hypo : el_a_sup app liste}

VAR cour,prec : d_el_lnom;
```

```
BEGIN
   IF el_a_sup = liste THEN
      BEGIN
         liste := liste^.next;
         DISPOSE(el_a_sup);
      END
   ELSE
      BEGIN
         cour := liste;
         WHILE (cour <> el_a_sup ) AND (cour <> NIL) DO
            BEGIN
               prec := cour;
               cour := cour^.next;
            END;
         IF cour = NIL THEN WRITELN(TTY,'CATASTROPHE')
            ELSE
               BEGIN
                  prec^.next := cour^.next;
                  DISPOSE(el_a_sup);
               END;
      END;
END;

{=============================================================================}

FUNCTION lnom_conv(VAR fich : f_el_nom) : d_el_lnom;

VAR liste_nom,endroit_ins : d_el_lnom;

BEGIN
   RESET(fich);
   liste_nom := lnom_cree;
   endroit_ins := VIDE;
   WHILE NOT EOF(fich) DO
      BEGIN
         lnom_ins(liste_nom,fich^,endroit_ins);
         endroit_ins := lnom_suivant(liste_nom,endroit_ins);
         GET(fich);
      END;
   lnom_conv := liste_nom;
END;

{-----------------------------------------------------------------------------}

FUNCTION lmc_conv(VAR fich : f_el_mc) : d_el_lmc;

VAR liste_mc,endroit_ins : d_el_lmc;

BEGIN
   RESET(fich);
   liste_mc := lmc_cree;
   endroit_ins := VIDE;
   WHILE NOT EOF(fich) DO
      BEGIN
         lmc_ins(liste_mc,fich^,endroit_ins);
         endroit_ins := lmc_suivant(liste_mc,endroit_ins);
         GET(fich);
      END;
   lmc_conv := liste_mc;
```

```
END;

{=============================================================================}

PROCEDURE lnom_search(deb_rech : d_el_lnom;nom_cherche : str20;
                     VAR d_nom_trouve : d_el_lnom);

{ cherche dans une liste de noms le nom "nom_cherche", a partir de l'endroit
  vers lequel pointe "deb_rech". "d_nom_trouve" pointe vers l'element
  correspondant s'il existe, ou vaut VIDE s'il n'existe pas. }


VAR courant : d_el_lnom;
    nom_cour,l_nom_cherche,l_nom_cour : str20;

BEGIN
   IF deb_rech = VIDE THEN d_nom_trouve := VIDE
      ELSE
         BEGIN
            lower(nom_cherche,l_nom_cherche);
            courant := deb_rech;
            lnom_content(courant,nom_cour);
            lower(nom_cour,l_nom_cour);
            WHILE (l_nom_cherche <> l_nom_cour) AND (courant <> VIDE) DO
               BEGIN
                  courant := lnom_suivant(deb_rech,courant);
                  IF courant <> VIDE THEN
                     BEGIN
                        lnom_content(courant,nom_cour);
                        lower(nom_cour,l_nom_cour);
                     END;
               END;
            d_nom_trouve := courant;
         END;
END;

{-----------------------------------------------------------------------------}

PROCEDURE lmc_simili_search(deb_rech : d_el_lmc;mc_cherche : str20;
                     VAR d_mc_trouve : d_el_lmc;VAR delta : INTEGER);

{ cherche dans une liste de mots-cles le premier mot-cle similaire a
  "mc_cherche", a partir de l'endroit vers lequel pointe "deb_rech".
  "d_mc_trouve" pointe vers l'element correspondant s'il existe,
   ou vaut VIDE s'il n'existe pas. }

VAR courant : d_el_lmc;
    mc_cour,l_mc_cherche,l_mc_cour : str20;

BEGIN
   IF deb_rech = VIDE THEN BEGIN delta := -1;d_mc_trouve := VIDE END
      ELSE
         BEGIN
            lower(mc_cherche,l_mc_cherche);
            courant := deb_rech;
            lmc_content(courant,mc_cour);
            lower(mc_cour,l_mc_cour);
            delta := simili(l_mc_cherche,l_mc_cour);
            WHILE (delta < 0) AND (courant <> VIDE) DO
               BEGIN
```

```
                courant := lmc_suivant(deb_rech,courant);
                IF courant <> VIDE THEN
                   BEGIN
                      lmc_content(courant,mc_cour);
                      lower(mc_cour,l_mc_cour);
                      delta := simili(l_mc_cherche,l_mc_cour);
                   END;
             END;
          d_mc_trouve := courant;
       END;
END;

{=============================================================================}
```

```
PROCEDURE aff_el_hlp(element : el_hlp);

VAR i : INTEGER;
    jolie_date : str20;

BEGIN
   WITH element DO
      BEGIN
         WRITELN(TTY,nom);
         WRITELN(TTY,fct);
         pretty_date(date_maj,jolie_date);
         WRITELN(TTY,jolie_date);
         i := 1;
         WHILE (i <= nb_max_mc) DO
            BEGIN
               IF mc[i] <> bl20 THEN BEGIN
                                        WRITELN(TTY,mc[i]);
                                        i := i + 1;
                                     END
                                ELSE i := nb_max_mc + 1; {pour finir la boucle}
            END;
      END;
END;

{=============================================================================}
```

```
PROCEDURE intro_nom (VAR existe : BOOLEAN);

VAR nom: str20;
    d_nom_trouve,d_nom_vide : d_el_lnom;
    nom_a_inserer : el_nom;

BEGIN
   d_nom_vide := NIL;

   WRITE(TTY,'Introduisez le nom : ');
   READLN(TTY);READ(TTY,nom);

   lnom_search(lnom,nom,d_nom_trouve);
   IF d_nom_trouve <> vide THEN
      BEGIN
         existe := TRUE;
         setpos(fhlp,d_nom_trouve^.adresse);   { get implicite }
         WRITELN(TTY,'Il existe deja un programme de ce nom,');
         WRITE(TTY,'dont voici les informations :');
         aff_el_hlp(fhlp^);
         setpos(fhlp,pos_append,TRUE);
         WRITELN(TTY,'aborting...');
         WRITELN(TTY);
      END
   ELSE
      BEGIN
         existe := FALSE;
         nom_a_inserer.content := nom;
         nom_a_inserer.adresse := pos_append;
         lnom_ins(lnom,nom_a_inserer,d_nom_vide);
         fhlp^.nom := nom;
      END;
END;

{-------------------------------------------------------------------------}

PROCEDURE intro_fonction;

BEGIN
   WRITELN(TTY,'Intro la fonction : ');
   READLN(TTY);READ(TTY,fhlp^.fct);
END;

{-------------------------------------------------------------------------}

PROCEDURE intro_date;

VAR date_in : str16;
    date_du_jour : str9;
    cod_ret : typ_code_ret;

BEGIN
   cod_ret := ok;
   WRITE(TTY,'Intro de la date (defaut : date du jour) : ');
   READLN(TTY);READ(TTY,date_in);
   IF date_in = bl16 THEN BEGIN
                             date_du_jour := date;
                             conv_date(date_du_jour,fhlp^.date_maj,cod_ret)
                          END
```

```
                    ELSE conv_date(date_in,fhlp^.date_maj,cod_ret);
   WHILE cod_ret <> ok DO
      BEGIN
         WRITELN(TTY,'Date erronnee');
         WRITELN(TTY,'Quelle date ? ');
         READLN(TTY);READ(TTY,date_in);
         IF date_in = bl16 THEN BEGIN
                                  date_du_jour := date;
                                  conv_date(date_du_jour,fhlp^.date_maj,cod_ret)
                                END
                           ELSE conv_date(date_in,fhlp^.date_maj,cod_ret);
      END;
END;

{-------------------------------------------------------------------------}

PROCEDURE intro_mc;

VAR mot_cle :str20;
    bidon,i,j,delta : INTEGER;
    double : BOOLEAN;
    endroit_cour,d_mc_trouve : d_el_lmc;
    c : CHAR;
    mc_a_inserer : el_mc;

BEGIN
   FOR i := 1 TO nb_max_mc DO fhlp^.mc[i] := bl20;
   i := 1;

   WRITELN(TTY,'Intro des mots-cles (max ',nb_max_mc:2,' ) ');
   WRITELN(TTY,'Introduisez ''Z'' pour terminer');

   WRITE(TTY,'Mot-cle ',i:2,' : ');
   READLN(TTY);READ(TTY,mot_cle);

   WHILE (mot_cle <> 'z                    ' ) DO
      BEGIN
         endroit_cour := lmc;

         lmc_simili_search(endroit_cour,mot_cle,d_mc_trouve,delta);
         WHILE delta > 0 DO
            BEGIN
               WRITE(TTY,'Le mot-cle "',mot_cle,'" est similaire au mot-cle "');
               WRITE(TTY,d_mc_trouve^.content,'" qui figure parmi les ');
               WRITELN(TTY,'mots-cles deja definis');
               WRITE(TTY,'Doivent-ils etre consideres comme egaux ? ');
               getc(c);WRITELN(TTY);
               IF confirm(c) THEN
                  BEGIN
                     mot_cle := d_mc_trouve^.content;
                     delta := 0 ; {pour terminer la boucle prematurement}
                  END
               ELSE
                  BEGIN
                     endroit_cour := d_mc_trouve^.next;
                     lmc_simili_search(endroit_cour,mot_cle,d_mc_trouve,delta);
                  END;
            END;
         mc_a_inserer.content := mot_cle;
         mc_a_inserer.adresse := pos_append;
```

```
            lmc_ins(lmc,mc_a_inserer,endroit_cour);

                        { on va faire une recherche pour les cas tordus ou un
                          mot-cle est devenu egal a un autre de la meme fiche par
                          le jeu des similarites acceptees comme egalite }
            double := FALSE;
            FOR j := 1 TO (i - 1) DO
               IF mot_cle = fhlp^.mc[i] THEN double := TRUE;
            IF NOT double THEN
               BEGIN
                  fhlp^.mc[i] := mot_cle;
                  i := i + 1 ;
               END;

            IF i < nb_max_mc THEN
               BEGIN
                  WRITE(TTY,'Mot-cle ',i:2,' : ');
                  READLN(TTY);READ(TTY,mot_cle);
               END
            ELSE
               mot_cle := 'z                    ' : {pour terminer la boucle}
         END;
END;

{-------------------------------------------------------------------------}

PROCEDURE intro_doc;

VAR ligne : str80;
    i,j : INTEGER;

BEGIN
   WRITELN(TTY,'Intro les "sources alternatives de documentation" (Max 3 lignes): ');
   i := 1;
   READLN(TTY);READ(TTY,ligne);
   WHILE (i <= 3) AND (ligne <> 'z                                                ' ) DO
      BEGIN
         fhlp^.docu[i] := ligne;
         i := i + 1;
         READLN(TTY);READ(TTY,ligne);
      END;

   FOR j := i TO 3 DO fhlp^.docu[i] := b180;

END;

{-------------------------------------------------------------------------}

PROCEDURE trt_ajout;

VAR existe : BOOLEAN;

BEGIN
   intro_nom(existe);
   IF NOT existe THEN
      BEGIN
         intro_fonction;
         intro_date;
         intro_mc;
         intro_doc;
```

```
            lmc_ins(lmc,mc_a_inserer,endroit_cour);

                     { on va faire une recherche pour les cas tordus ou un
                       mot-cle est devenu egal a un autre de la meme fiche par
                       le jeu des similarites acceptees comme egalite }
            double := FALSE;
            FOR j := 1 TO (i - 1) DO
               IF mot_cle = fhlp^.mc[i] THEN double := TRUE;
            IF NOT double THEN
               BEGIN
                  fhlp^.mc[i] := mot_cle;
                  i := i + 1 ;
               END;

            IF i < nb_max_mc THEN
               BEGIN
                  WRITE(TTY,'Mot-cle ',i:2,' : ');
                  READLN(TTY);READ(TTY,mot_cle);
               END
            ELSE
               mot_cle := 'z                    ' ; {pour terminer la boucle}
         END;
END;

{-------------------------------------------------------------------------}

PROCEDURE intro_doc;

VAR ligne : str80;
    i,j : INTEGER;

BEGIN
   WRITELN(TTY,'Intro les "sources alternatives de documentation" (Max 3 lignes): ');
   i := 1;
   READLN(TTY);READ(TTY,ligne);
   WHILE (i <= 3) AND (ligne <> 'z                                                ' ) DO
      BEGIN
         fhlp^.docu[i] := ligne;
         i := i + 1;
         READLN(TTY);READ(TTY,ligne);
      END;

   FOR j := i TO 3 DO fhlp^.docu[i] := b180;

END;

{-------------------------------------------------------------------------}

PROCEDURE trt_ajout;

VAR existe : BOOLEAN;

BEGIN
   intro_nom(existe);
   IF NOT existe THEN
      BEGIN
         intro_fonction;
         intro_date;
         intro_mc;
         intro_doc;
```

```
            PUT(fhlp);
            pos_append := curpos(fhlp);
         END;

END; {procedure trt_ajout}

{=============================================================================}
```

```
PROCEDURE maj_nom(VAR existe : BOOLEAN);

VAR nom : str20;
    d_nom_trouve : d_el_lnom;

BEGIN
   WRITE(TTY,'Introduisez le nom : ');
   READLN(TTY);READ(TTY,nom);

   lnom_search(lnom,nom,d_nom_trouve);
   IF d_nom_trouve = vide THEN
      BEGIN
         existe := FALSE;
         WRITELN(TTY,'Pas de fiche correspondante');
         WRITELN(TTY,'Aborting...');
      END
   ELSE
      BEGIN
         existe := TRUE;
         setpos(fhlp,d_nom_trouve^.adresse);   { get implicite }
      END;
END;

{-------------------------------------------------------------------------}

PROCEDURE maj_fonction;

VAR fonction : str80;
    count : INTEGER;

BEGIN
   WRITE(TTY,'Introduisez la fonction (defaut : ''');
   wstr(TTY,fhlp^.fct,count);WRITELN(TTY,''')');
   READLN(TTY);READ(TTY,fonction);
   IF fonction <> b180 THEN fhlp^.fct := fonction;
END;

{-------------------------------------------------------------------------}

PROCEDURE maj_date;

VAR jolie_date : str20;
    date_in : str16;
    cod_ret : typ_code_ret;
    date_du_jour : str9;
    date_int : str10;

BEGIN
   cod_ret := ok;
   WRITE(TTY,'Introduisez la date (defaut : ''');
   date_du_jour := date;
   conv_date(date_du_jour,date_int,cod_ret);
   pretty_date(date_int,jolie_date);
   WRITE(TTY,jolie_date);WRITELN(TTY,''')');
   READLN(TTY);READ(TTY,date_in);
   IF date_in <> b116 THEN conv_date(date_in,fhlp^.date_maj,cod_ret);

   WHILE cod_ret <> ok DO
      BEGIN
```

```
            WRITELN(TTY,'Date erronnee');
            WRITELN(TTY,'Ouelle date ? ');
            READLN(TTY);READ(TTY,date_in);
            IF date_in <> bl16 THEN conv_date(date_in,fhlp^.date_maj,cod_ret);
         END;
END;

{-------------------------------------------------------------------------}

PROCEDURE mai_mc;

VAR mot_cle :str20;
    bidon,i,j,delta : INTEGER;
    double : BOOLEAN;
    endroit_cour,d_mc_trouve,d_ancien_mc : d_el_lmc;
    c : CHAR;
    mc_a_inserer : el_mc;
    count : INTEGER;

BEGIN
   WRITELN(TTY,'Introduction des mots-cles (max ',nb_max_mc:2,' ) : ');
   WRITELN(TTY,'Introduisez ''Z'' pour terminer ');
   WRITELN(TTY);
   i := 1;

   WRITE(TTY,'Mot-cle ',i:2,' (Defaut : ''');
   wstr(TTY,fhlp^.mc[i],count);
   WRITE(TTY,''') : ');
   READLN (TTY);READ(TTY,mot_cle);

   WHILE (mot_cle <> 'z                   ') DO
      BEGIN
         IF mot_cle = bl20 THEN i := i + 1
                           { si le mc est blanc on ne fait rien et on
                           garde donc l'ancienne valeur }
      ELSE
         BEGIN
            IF fhlp^.mc[i] <> bl20 THEN
               BEGIN
                  lmc_simili_search(lmc,fhlp^.mc[i],d_ancien_mc,delta);
                  IF delta <> 0 THEN WRITELN(TTY,'CATASTROPHE, ON ESSAIE DE METTRE A JOUR UN MOT-CLE INEXISTANT')
                           ELSE lmc_suppress(lmc,d_ancien_mc);
               END;
            WRITELN(TTY,'Mot-cle introduit : ',mot_cle);
            endroit_cour := lmc;
            lmc_simili_search(endroit_cour,mot_cle,d_mc_trouve,delta);
            WHILE delta > 0 DO
               BEGIN
                  WRITE(TTY,'Le mot-cle "',mot_cle,'" est similaire au mot-cle "');
                  WRITE(TTY,d_mc_trouve^.content,'" qui figure parmi les ');
                  WRITELN(TTY,'mots-cles deja definis');
                  WRITE(TTY,'Doivent-ils etre consideres comme egaux ? ');
                  getc(c);WRITELN(TTY);
                  IF confirm(c) THEN
                     BEGIN
                        mot_cle := d_mc_trouve^.content;
                        delta := 0 ; {pour terminer la boucle}
                     END
                  ELSE
                     BEGIN
```

```
                READLN(TTY);READ(TTY,ligne);
              END;
            FOR j := i TO 3 DO fhlp^.docu[i] := bl80;
        END;
END;

{-------------------------------------------------------------------------}

PROCEDURE trt_maj;

VAR existe : BOOLEAN;

BEGIN
   maj_nom(existe);
   IF existe THEN
      BEGIN
         maj_fonction;
         maj_date;
         maj_mc;
         maj_docu;

         putx(fhlp);
         setpos(fhlp,pos_append);
      END;

END; {procedure  trt_maj}

{=========================================================================}
```

```
PROCEDURE trt_suppr;

LABEL 1;

VAR nom: str20;
    d_nom_trouve : d_el_lnom;
    rep : str3;
    i,delta : INTEGER;
    d_mc_trouve : d_el_lmc;
    debut_de_rech : d_el_lmc;

BEGIN
   WRITELN(TTY,'Introduisez le nom du programme dont vous desirez supprimer la fiche ');
   READLN(TTY);READ(TTY,nom);
   lnom_search(lnom,nom,d_nom_trouve);
   IF d_nom_trouve = NIL THEN
      BEGIN
         WRITELN(TTY,'ERREUR ! Pas de programme de ce nom !');
         WRITELN(TTY,'Aborting...');
         GOTO 1;
      END;
   setpos(fhlp,d_nom_trouve^.adresse);
   WRITELN(TTY,'Voici la fiche a supprimer : ');
   aff_el_hlp(fhlp^);
   REPEAT
      WRITELN(TTY,'Confirmez-vous la suppression ?');
      READLN(TTY);READ(TTY,rep);
   UNTIL (rep = 'oui') OR (rep = 'non');
   IF rep = 'non' THEN
      BEGIN
         WRITELN(TTY,'Aborting...');
         GOTO 1 ;
      END;

   i := 1;

   debut_de_rech := lmc_suivant(lmc,VIDE);
   WHILE i <= nb_max_mc DO
      BEGIN
         IF fhlp^.mc[i] <> b120
            THEN
               BEGIN
                  lmc_simili_search(debut_de_rech,fhlp^.mc[i],d_mc_trouve,delta);
                  IF d_nom_trouve^.adresse = d_mc_trouve^.adresse THEN
                     BEGIN
                        lmc_suppress(lmc,d_mc_trouve);
                        i := i + 1;
                     END
                  ELSE
                     debut_de_rech := lmc_suivant(lmc,d_mc_trouve);
               END
            ELSE i := nb_max_mc + 1;
      END;

   lnom_suppress(lnom,d_nom_trouve);

   fhlp^.nom := maxstr20;
   putx(fhlp);
1: setpos(fhlp,pos_append);
```

```
END; {procedure trt_supp}

{=============================================================================}
```

```
PROCEDURE fnom_sort(VAR fich : f_el_nom);EXTERN;

PROCEDURE fdate_sort(VAR fich : f_el_date);EXTERN;

PROCEDURE fmc_sort(VAR fich : f_el_mc);EXTERN;


{ Programme principal }
{-------------------------------------------------------------------------------}
BEGIN

   WRITE(TTY,'Introduction ? ');
   REPEAT
      READLN(TTY);READ(TTY,rep);
   UNTIL (rep = 'oui') OR (rep = 'non');
   WRITELN(TTY);
   IF rep = 'non' THEN GOTO 100;

   RESET(fnom,'fnom.hlp','/O');
   IF erstat(fnom) <> 0 THEN lnom := VIDE
                        ELSE lnom := lnom_conv(fnom);
   RESET(fmc,'fmc.hlp','/O');
   IF erstat(fmc) <> 0 THEN lmc := VIDE
                       ELSE lmc := lmc_conv(fmc);

   append(fhlp,'fhlp.hlp','/O');
   IF erstat(fhlp) <> 0 THEN BEGIN
                               REWRITE(fhlp,'fhlp.hlp');
                               pos_append := 0;
                             END
                        ELSE pos_append := curpos(fhlp);

update(fhlp,'fhlp.hlp');
setpos(fhlp,pos_append,TRUE);

WRITELN(TTY,'Que desirez-vous faire ?');
WRITELN(TTY,'   Ajout');
WRITELN(TTY,'   Mise-a-jour');
WRITELN(TTY,'   Suppression');
WRITELN(TTY,'   Quitter');
READLN(TTY);READ(TTY,choix);
lower(choix,l_choix);

WHILE NOT match(l_choix,'quitter') DO
   BEGIN
      IF match(l_choix,'ajout') THEN trt_ajout
      ELSE IF match(l_choix,'mise-a-jour') THEN trt_maj
      ELSE IF match(l_choix,'suppression') THEN trt_supp
      ELSE bell;

      WRITELN(TTY,'Que desirez-vous faire ?');
      WRITELN(TTY,'   Ajout');
      WRITELN(TTY,'   Mise-a-jour');
      WRITELN(TTY,'   Suppression');
      WRITELN(TTY,'   Quitter');
      READLN(TTY);READ(TTY,choix);
      lower(choix,l_choix);
   END;
```

```
RESET(fhlp,'fhlp.hlp','/I');
REWRITE(ftemp);
REWRITE(fnom,'fnom.hlp');
REWRITE(fmc,'fmc.hlp');
REWRITE(fdate,'fdate.hlp');

posit := curpos(fhlp);
GET(fhlp);
WHILE NOT EOF(fhlp) DO
   BEGIN
      IF fhlp^.nom <> maxstr20 THEN
         BEGIN
            ftemp^ := fhlp^;
            PUT(ftemp);

            fnom^.content := fhlp^.nom;
            fnom^.adresse := posit;
            PUT(fnom);

            fdate^.content := fhlp^.date_maj;
            fdate^.adresse := posit;
            PUT(fdate);

            i := 1;
            WHILE (i <= nb_max_mc) DO
               BEGIN
                  IF fhlp^.mc[i] <> b120 THEN
                     BEGIN
                        fmc^.content := fhlp^.mc[i];
                        fmc^.adresse := posit;
                        PUT(fmc);
                        i := i + 1;
                     END
                  ELSE i := nb_max_mc + 1;
                        {pour sortir du while}
               END;
         END;
      posit := curpos(fhlp);
      GET(fhlp);
   END;
rename(ftemp,'fhlp.hlp');

WRITE(TTY,'tri ? ');
REPEAT
   READLN(TTY);READ(TTY,rep);
UNTIL (rep = 'oui') OR (rep = 'non');
WRITELN(TTY);
IF rep = 'non' THEN GOTO 101;

100:   WRITE(TTY,'Tri du fichier des noms : ');
fnom_sort(fnom);
WRITELN(TTY,'OK');
WRITE(TTY,'Tri du fichier des dates : ');
fdate_sort(fdate);
WRITELN(TTY,'OK');
WRITE(TTY,'Tri du fichier des mot-cles : ');
fmc_sort(fmc);
WRITELN(TTY,'OK');
101:
```

```
(*$M-*)       { pas de programme principal }

{ copyright 1984 by Charles BOKOR and 'Centre de Calcul' of Facultes
  universitaires Notre Dame de la Paix, Namur, Belgium }

{ Ceci constitue les routines d'un gestionnaire d'ecran 'universel', dans
  la mesure ou il est adaptable a tout terminal video. Les idees de
  depart sont empruntees a CURSES ( gestionnaire d'ecran ecrit en C).

  Ce programme a ete ecrit et mis au point en fevrier/mars 1984 par
  Charles BOKOR, dans le cadre d'un memoire de fin d'etudes de licence
  et maitrise en informatique visant a creer un systeme de
  documentation on line interactif a interface utilisateur evolue.}
```

GETED1

{ Dans l'etat actuel des choses, ce gestionnaire peut etre divise en
trois couches :
- Une serie de routines fournissant des services de base au niveau du terminal physique (mouvement de curseur, effacement d'ecran). Ces routines sont reprises dans le module TSCR.
- Des routines (que l'on trouvera ci-apres), qui travaillent au niveau d'un ecran logique.
- Une troisieme couche, a l'etat embryonnaire, travaille au niveau 'semantique, c'est a dire qu'il manipule des objets de type particulier : messages, ecrans de menu (a implementer), grilles de saisie (a implementer),...}

```
include 'sys:string.dec';     { definition de strings de diverse longueur et
                                des constantes 'string blanc' associees }

include 'sys:utilit.dec';     { declaration de diverses procedures et fonctions
                                utilitaires }

include 'sys:cod_ret.dec';    { definitions des constantes et types pour l'
                                affichages des messages d'erreur ou de
                                confirmation}

include 'sys:tscr.dec';       { definitions des constantes,types et procedures
                                necessaires pour la gestion de l'ecran du
                                terminal physique }


TYPE {typ_rendition = (normal,bold,blink,reverse,under);defini ailleurs}
     ar_el = PACKED ARRAY [1..lim_lin,1..lim_col] OF CHAR ; { representation
                                                              du contenu
                                                              d'une fenetre }
     ar_car = PACKED ARRAY [1..lim_lin,1..lim_col] OF typ_rendition;
                                { representation du type d'affichage }
          { logiquement, on devrait associer chaque caractere avec son
            "rendition", p.ex. au sein d'un record. On ne l'a pas fait pour
            des questions d'efficacite de stockage des fenetres sur disque }

     st_window = PACKED RECORD  { representation d'une fenetre }
        curx,cury : INTEGER;
        begx,begy : INTEGER;
        maxx,maxy : INTEGER;
        clear : BOOLEAN;
        content : ar_el;
        rendition : ar_car;
        ar_mod : PACKED ARRAY [1..lim_lin] OF BOOLEAN;
        nom_win : str10;
      END;

      window = ^st_window;

      f_window = FILE OF st_window;


VAR msgline_1 : INTEGER;                  { lignes de l'ecran ou s'afficheront}
    msgline_2 : INTEGER;                  { les divers messages }

    curscr : window;

    code_ret : typ_code_ret;
    t : tab_msg_ret;                      { table des messages }

    ligne_blanche : PACKED ARRAY [1..lim_col] OF CHAR;

    fimpr : TEXT;                         { spooler (spouleur !) d'impression }

{-------------------------------------------------------------------------------}
```

```
{ Les fonctions qui vont suivre sont des fonctions qui implementent
  l'"interface abstrait" des "fenetres" }

FUNCTION begx(curwin : window) : INTEGER;
BEGIN
   begx := curwin^.begx;
END;

FUNCTION begy(curwin : window) : INTEGER;
BEGIN
   begy := curwin^.begy;
END;

FUNCTION maxx(curwin : window) : INTEGER;
BEGIN
   maxx := curwin^.maxx;
END;

FUNCTION maxy(curwin : window) : INTEGER;
BEGIN
   maxy := curwin^.maxy;
END;

FUNCTION curx(curwin : window) : INTEGER;
BEGIN
   curx := curwin^.curx;
END;

FUNCTION cury(curwin : window) : INTEGER;
BEGIN
   cury := curwin^.cury;
END;

FUNCTION content(curwin : window;x,y : INTEGER):CHAR;
BEGIN
   content := curwin^.content[x,y];
END;

FUNCTION rendition(curwin : window;x,y : INTEGER):typ_rendition;
BEGIN
   rendition := curwin^.rendition[x,y];
END;

PROCEDURE addchar(VAR curwin : window; c: CHAR; rend : typ_rendition);

{ IN : curwin : designateur d'une fenetre
       c : caractere
       rend : mode d'affichage video
  OUT : fenetre modifiee
  semantique : ajoute le caractere "c",en mode d'affichage "rend",a la
               position courante de la fenetre "curwin". La position
               courante en "y" est avancee d'une position.
}
BEGIN
   WITH curwin^ DO
      BEGIN
         content[curx,cury] := c;
         rendition[curx,cury] := rend;
```

```
            curv := curv + 1;
            ar_mod[curx] := TRUE;
        END;
END;

{=============================================================================}
```

```
{ Procedures d'initialisation et de creation }

PROCEDURE move(VAR curwin : window;x,y : INTEGER);FORWARD;

{ IN : curwin : designateur d'une fenetre
       x,y : position dans la fenetre
  OUT : fenetre modifiee
  semantique : modifie la position courante de "curwin" en "x","y"
}

{-------------------------------------------------------------------------------}

PROCEDURE init_fields(VAR curwin : window;all_screen : BOOLEAN );

{Window dependent}

{ IN : curwin : designateur vers une fenetre
       all_screen : booleen
  OUT : fenetre modifiee
  semantique : initialise les differentes composantes de la representation
               d'une fenetre;
               Si le booleen "all-screen" est vrai, l'initialisation se fait
               sur toute la representation physique de la fenetre ( qui
               n'est effectivement utilisee que dans le cas d'une fenetre
               de taille maximale).Sinon, l'initialisation ne se fait que
               sur la partie de la representation physique qui sert
               effectivement a representer la fenetre
}

VAR i,j : INTEGER;
    debx,deby,finx,finy : INTEGER;

BEGIN
   IF all_screen THEN BEGIN
                        debx := 1 ; finx := xphysx;
                        deby := 1 ; finy := xphysy;
                      END
                 ELSE BEGIN
                        debx := begx(curwin); finx := maxx(curwin);
                        deby := begy(curwin); finy := maxy(curwin);
                      END;
   FOR i := debx TO finx DO
      BEGIN
         curwin^.ar_mod[i] := FALSE;
         FOR j:= deby TO finy DO
            BEGIN
               move(curwin,i,j);
               addchar(curwin,' ',normal);
            END
      END;
   curwin^.curx := debx;curwin^.cury := deby;
END ;

{-------------------------------------------------------------------------------}

FUNCTION newwin(homex,homey,edgex,edgey : INTEGER):window;

{ window dependent }
```

```
{ IN : homex,homey,edgex,edgey : coordonnees des quatres coins d'une fenetre
  valeur de la fonction : designateur d'une fenetre
  semantique : cree la representation d'une fenetre debutant a "homex",
  "homey", dont le coin inferieur droit est place en "edgex","edgey" ,
  et retourne un pointeur vers cette representation
}

VAR created_window : window;

BEGIN
   NEW (created_window);
   newwin := created_window;
   WITH created_window^ DO
      BEGIN
         IF homex > 0 THEN begx := homex ELSE begx := 0;
         IF homey > 0 THEN begy := homey ELSE begy := 0;
         IF edgex <= xphysx THEN maxx := edgex ELSE maxx := xphysx;
         IF edgey <= xphysy THEN maxy := edgey ELSE maxy := xphysy;
         curx := homex ; cury := homey ;
         clear := FALSE;
      END;
   init_fields(created_window,TRUE)
END;

{-------------------------------------------------------------------------}

PROCEDURE set_msgline(i :INTEGER);

{ IN : i : entier
  parametres globaux : msgline_1, msgline_2 : lignes de l'ecran ou devront
                                              etre affichees les messages
  semantique : definit les 2 lignes de l'ecran ou seront affiches les
               messages ( "i" : premiere ligne de messages )
}


BEGIN
   IF i > (xphysx - 1) THEN i := xphysx - 1;
   msgline_1 := i;
   msgline_2 := i + 1;
END;

{-------------------------------------------------------------------------}

PROCEDURE initscr(VAR code_ret : typ_code_ret);

{ IN :
  OUT : code_ret : code_retour
  parametres globaux : fichier des codes retour, de nom "nom_fich_msg"
                       ligne_blanche : chaine de caracteres, de longueur
                                       egale a la largeur maximale de
                                       l'ecran, garnie de blancs
                       curscr : designateur de la fenetre representant
                                l'aspect de l'ecran du terminal au moment
                                courant
  semantique : initialise les differentes choses necessaires pour la
               gestion d'ecran .
               Le code de retour vaut 'ok' en cas de reussite
               En cas d'erreur,l'initialistion n'est pas faite; (en
               particulier, il n'est pas possible d'employer des routines
```

```
                faisant appel a la gestion d'ecran pour afficher les
                messages d'erreur

        Valeur des codes-retour :
            ok
            fcdrep : fichier des codes-retour n'existe pas
            fecrep : fichier de description des ecrans n'existe pas
            tnd : terminal non defini (pas de nom logique TERM:)
            tinc : terminal inconnu (pas de description)

}

LABEL 100;

VAR fich_ret : TEXT;
    i : INTEGER;
    cod_ret_cour,ind_cod_ret : typ_code_ret;

BEGIN
   code_ret := ok;

    {lecture des messages }

   RESET(fich_ret,nom_fich_msg,'/O');
   IF erstat(fich_ret) <> 0 THEN
      BEGIN
         code_ret := fcdrep;
         GOTO 100
      END;
   cod_ret_cour := ok;
   WHILE NOT EOF(fich_ret) AND (cod_ret_cour <= fin_cod_sans_msg) DO
      BEGIN
         READ(fich_ret,t[cod_ret_cour]);READLN(fich_ret);
         cod_ret_cour := SUCC(cod_ret_cour);
      END;
   FOR ind_cod_ret := cod_ret_cour TO fin_cod_sans_msg DO
                           t[ind_cod_ret] := val_init_t_cod_ret;

   FOR i := 1 TO lim_col DO ligne_blanche[i] := ' ';

   xiscr(code_ret);
   xcls;
   IF code_ret <> ok THEN GOTO 100;

   set_msgline(xphysx - 1 );

   curscr := newwin(1,1,xphysx,xphysy);
100:

END {initscr};

{-------------------------------------------------------------------------}

PROCEDURE endscr;

{ IN :
  OUT :
  parametre global : fichier interne "spooler" des impressions, de nom interne
                     "fimpr"
  semantique : termine la gestion d'ecran et imprime le fichier d'impression
```

```
}

VAR fout : TEXT;

BEGIN
   RESET(fimpr,'','/E/O');
   IF erstat(fimpr) = 0 THEN
      BEGIN
         REWRITE(fout,'lpt0:imprim.txt');
         WHILE NOT EOF(fimpr) DO
            BEGIN
               fout^ := fimpr^;
               PUT(fout);
               GET(fimpr);
            END;
      END;
   xcls;
   xrscr;
END;

{=============================================================================}
```

```
{ Procedures agissants sur une 'fenetre }


PROCEDURE move;

{ cf supra }

BEGIN
   curwin^.curx := x;
   curwin^.cury := y;
END;

{-------------------------------------------------------------------------}

PROCEDURE refresh(VAR curwin : window);


{ window dependent !!!!!!!!!! (pour des raisons de rapidite)}

{ IN : curwin : designateur d'une fenetre
  OUT : action sur l'ecran du terminal
  parametres globaux : curscr : designateur de la fenetre representant
                                l'aspect de l'ecran du terminal au moment
                                courant
  semantique : synchronise l'ecran physique avec le contenu de la fenetre
               "curwin" tel que le connait le gestionnaire d'ecran
}

VAR bidon,longueur_max,i,j : INTEGER;
    pos_maxx,pos_maxy : INTEGER;
    refresh_force : BOOLEAN;
    cur_rend : typ_rendition;
    x_phys,y_phys : INTEGER;

BEGIN
   x_phys := -1 ;
   y_phys := -1 ;
   cur_rend := normal;
   IF curwin = curscr THEN refresh_force := TRUE
                      ELSE refresh_force := FALSE;

   WITH curwin^ DO
    BEGIN

      IF maxx <= xphysx THEN pos_maxx := maxx
                        ELSE pos_maxx := xphysx;
      IF maxy <= xphysy THEN pos_maxy := maxy
                        ELSE pos_maxy := xphysy;

      IF clear OR refresh_force THEN
         BEGIN
            xclear(begx,begy,maxx,maxy);
            IF clear THEN
               BEGIN
                  clear := FALSE;
                  init_fields(curscr,FALSE);
               END;
         END {if clear ou refresh_force};
```

```
FOR i := beax TO  pos_maxx DO
   BEGIN
      IF ar_mod[i] OR refresh_force THEN
         BEGIN
            FOR j := beay TO pos_maxv DO
               BEGIN
                  IF (refresh_force AND (content[i,j] <> ' ')) OR
                     (curscr^.content[i,j] <> content[i,j]) OR
                     (curscr^.rendition[i,j] <> rendition[i,j])
                  THEN
                     BEGIN
                        IF (x_phys <> i) OR  (y_phys <> j) THEN
                           BEGIN
                              xmove(i,j);
                              x_phys := i;y_phys := j;
                           END;
                        IF (rendition[i,j] <> cur_rend) AND (rendition[i,j] <> normal) THEN
                           BEGIN
                              xso(rendition[i,j]);
                              cur_rend := rendition[i,j];
                           END;
                        IF (rendition[i,j] = normal) AND (cur_rend <> normal) THEN
                           BEGIN
                              xse;
                              cur_rend := normal;
                           END;
                        WRITE(TTY,content[i,j]);
                        y_phys := y_phys+1;
                        curscr^.content[i,j] := content[i,j];
                        curscr^.rendition[i,j] := rendition[i,j];
                     END {if};
                  END {boucle for j};
               ar_mod[i] := FALSE;
            END {if ligne modifiee};
         END;{boucle for i}
      IF cur_rend <> normal THEN xse;
      curscr^.curx := curx;
      curscr^.cury := cury;
      xmove(curx,cury);
   END {with}
END {procedure refresh};

{-------------------------------------------------------------------------------}

PROCEDURE l_refresh(x,y : INTEGER);

{ window dependent}

{ IN : x,y : position dans une fenetre
  OUT : action sur l'ecran du terminal.
  parametre global : curscr : designateur de la fenetre representant
                              l'aspect de l'ecran du terminal au moment
                              courant

  semantique :  synchronise la ligne "x", a partir de la position "y",
                avec cette partie de CURSCR
}

VAR cur_rend : typ_rendition;
    i : INTEGER;
```

```
    x_phys,y_phys : INTEGER;

BEGIN
   x_phys := -1;
   y_phys := -1;
   WITH curscr^ DO
      BEGIN
         xmove(x,y);
         xcli;
         cur_rend := normal;
         FOR j := y TO yphysy DO
            IF content[x,j] <> ' ' THEN
               BEGIN
                  IF (x_phys <> x) OR  (y_phys <> j) THEN
                     BEGIN
                        xmove(x,j);
                        x_phys := x;y_phys := j;
                     END;
                  IF (rendition[x,j] <> normal) AND (rendition[x,j] <> cur_rend ) THEN
                     BEGIN
                        xso(rendition[x,j]);
                        cur_rend := rendition[x,j];
                     END;
                  IF (rendition[x,j] = normal) AND (cur_rend <> normal) THEN
                     BEGIN
                        xse;
                        cur_rend := normal;
                     END;
                  WRITE(TTY,content[x,j]);
                  y_phys := y_phys+1;
               END {if et for};
            IF cur_rend <> normal THEN xse;
      END {with}
END { l_refresh };

{-------------------------------------------------------------------------------}

PROCEDURE touch(VAR curwin : window);

{ window dependent }

{ IN : curwin : designateur d'une fenetre.
  semantique : marque toutes les positions de l'ecran "curwin" comme
               ayant ete modifiees ;
               ceci implique qu'au prochain refresh, tout l'ecran sera
               reecrit
}

VAR i :INTEGER;

BEGIN
   WITH curwin^ DO
         FOR i := begx TO maxx DO ar_mod[i] := TRUE;
END;

{===============================================================================}
```

```
PROCEDURE message(VAR texte_1 : PACKED ARRAY [m..n:INTEGER] OF CHAR ;
                  VAR texte_2 : PACKED ARRAY [p..q:INTEGER] OF CHAR ;
                                VAR c: CHAR );

{ IN : texte_1, texte_2 : chaines de caracteres
  OUT : action sur l'ecran du terminal.
        c : caractere.
  parametres globaux : msgline_1, msgline_2 : lignes de l'ecran ou doivent
                                              s'afficher les messages
                       curscr : fenetre representant ce qui est affiche
                                au moment courant sur l'ecran du terminal.
  semantique : affiche un message de 2 lignes, en mode normal, sans
               modifier le contenu de l'ecran courant.
               Le message reste affiche jusqu'a l'introduction d'un
               caractere quelconque, qui est par apres disponible en "c"
}

VAR longueur_max,x,y : INTEGER;
    prem_ligne,deux_ligne : BOOLEAN;

BEGIN
   longueur_max := lim_col - 1;
   IF n - m + 1 = 0 THEN prem_ligne := FALSE ELSE prem_ligne := TRUE;
   IF q - p + 1 = 0 THEN deux_ligne := FALSE ELSE deux_ligne := TRUE;
   IF prem_ligne THEN
      BEGIN
         xmove(msgline_1,1);
         xcli;
         WRITE(TTY,texte_1:longueur_max);
      END;
   IF deux_ligne THEN
      BEGIN
         xmove(msgline_2,1);
         xcli;
         WRITE(TTY,texte_2:longueur_max);
      END;
   getc(c);
   IF prem_ligne THEN l_refresh(msgline_1,1);
   IF deux_ligne THEN l_refresh(msgline_2,1);
   xmove(curscr^.curx,curscr^.cury);
END;

{-------------------------------------------------------------------------------}

PROCEDURE erreur(VAR texte_1 : PACKED ARRAY [m..n:INTEGER] OF CHAR ;
                 VAR texte_2 : PACKED ARRAY [p..q:INTEGER] OF CHAR ;
                 VAR c: CHAR );

{ IN : texte_1, texte_2 : chaines de caracteres
       c : caractere
  OUT : affichage au terminal
  semantique : affiche un message d'erreur de 2 lignes, avec la premiere
               ligne en video inverse et la seconde en mode normal,
               et actionne le beeper du terminal.
               Le message reste affiche jusqu'a l'introduction d'un
               caractere quelconque, qui est par apres disponible en "c" }

VAR longueur_max,x,y : INTEGER;
```

```
    prem_ligne,deux_ligne : BOOLEAN;

BEGIN
   longueur_max := lim_col - 1;
   IF n - m + 1 = 0 THEN prem_ligne := FALSE ELSE prem_ligne := TRUE;
   IF a - b + 1 = 0 THEN deux_ligne := FALSE ELSE deux_ligne := TRUE;
   IF prem_ligne THEN
      BEGIN
         xso(reverse);
         bell;
         xmove(msgline_1,1);
         xcli;
         WRITE(TTY,texte_1:longueur_max);
         xse;
      END;
   IF deux_ligne THEN
      BEGIN
         xmove(msgline_2,1);
         xcli;
         WRITE(TTY,texte_2:longueur_max);
      END;
   getc(c);
   IF prem_ligne THEN l_refresh(msgline_1,1);
   IF deux_ligne THEN l_refresh(msgline_2,1);
   xmove(curscr^.curx,curscr^.cury);
END;

{-------------------------------------------------------------------------------}
```

```
{Fonctions du troisieme niveau ('semantique) }

PROCEDURE aff_2msg(msg_1, msg_2 : typ_code_ret; VAR c : CHAR);

{ IN : msg_1, msg_2 : code retour
  OUT : affichage au terminal
        c : caractere
  parametre global : t : table reprenant dans l'ordre les messages associes
                         aux codes_retour definis.
  semantique : affiche les messages correspondants aux codes-retour "msg_1"
               et "msg_2" sur les lignes destinees aux messages.
               Garde les messages affiches jusqu'a l'introduction d'un
               caractere, disponible en "c"
}

BEGIN
   message(t[msg_1], t[msg_2],c);
END;

{-------------------------------------------------------------------------}

PROCEDURE aff_2err(msg_1, msg_2 : typ_code_ret;VAR  c : CHAR);

{ IN : msg_1, msg_2 : code retour
  OUT : affichage au terminal
        c : caractere
  parametre global : t : table reprenant dans l'ordre les messages associes
                         aux codes_retour definis.
  semantique : affiche les messages correspondants aux codes-retour "msg_1"
               et "msg_2" sur les lignes destinees aux messages, le premier
               en video inverse et le second en mode normal.
               Garde les messages affiches jusqu'a l'introduction d'un
               caractere, disponible en "c"
}

BEGIN
   erreur(t[msg_1],t[msg_2],c);
END;

{-------------------------------------------------------------------------}

PROCEDURE aff_msg(msg : typ_code_ret);

{ IN : msg : code retour
  OUT : affichage au terminal
  parametre global : t : table reprenant dans l'ordre les messages associes
                         aux codes_retour definis.
  semantique : affiche le message correspondant au code retour "code_ret"
               sur la ligne destinee aux messages, et un message appropriee
               a la nature du premier sur la deuxieme ligne.
               Garde le message affiche jusqu'a l'introduction d'un
               caractere
}

VAR c : CHAR;

BEGIN
   IF (msg < deb_cod_avec_msg ) OR (msg > fin_cod_sans_msg )
     THEN aff_2err(minc,calguru,c)
```

```
    ELSE IF msg < fin_err
            THEN aff_2err(msg,mic,c)
            ELSE IF msg = mic
                    THEN aff_2msg(rien,mic,c)
                    ELSE IF (msg > fin_err) AND (msg < fin_msg)
                            THEN aff_2msg(msg,mic,c);
END;

{-------------------------------------------------------------------------}

FUNCTION conf_msg(msg : typ_code_ret): BOOLEAN;

{ IN : msg : code retour
  OUT : affichage au terminal
  valeur de la fonction : booleen
  parametre global : t : table reprenant dans l'ordre les messages associes
                         aux codes_retour definis.
  semantique : affiche le message correspondant au code retour "msg" sur
               la premiere ligne destinee aux messages,ainsi qu'un message
               de demande de confirmation sur la seconde ligne.
               Garde le message affiche jusqu'a l'introduction d'un
               caractere.
               Retourne TRUE si ce caractere est 'o' ou 'O' ou 'y' ou 'Y' ,
                        FALSE sinon .
}

VAR c : CHAR;

BEGIN
   IF (msg < deb_cod_avec_msg ) OR (msg > fin_cod_sans_msg )
     THEN aff_2err(minc,calculu,c)
     ELSE IF msg = dco THEN aff_2msg(rien,dco,c)
                  ELSE IF msg <= fin_msg
                          THEN aff_2err(paconf,rien,c)
                          ELSE aff_2msg(msg,dco,c);

   conf_msg := confirm(c);
END;

{-------------------------------------------------------------------------}

PROCEDURE ecr_msg(msg : typ_code_ret);

{ IN : msg : code retour
  OUT : affichage au terminal
  parametre global : t : table reprenant dans l'ordre les messages associes
                         aux codes_retour definis.
  semantique : affiche les messages correspondant au code retour "code_ret"
               sans faire appel a la gestion d'ecran.Cette procedure peut
               etre utilisee pour afficher les messages lorsque
               l'initialisation ne reussit pas ou hors d'un contexte de
               gestion d'ecran.
}

BEGIN
   IF (msg < deb_cod_avec_msg ) OR (msg > fin_cod_sans_msg)
      THEN WRITELN(TTY,t[minc])
      ELSE WRITELN(TTY,t[msg]);
END;
```

```
PROCEDURE addstr(VAR curwin : window;
                 VAR string : PACKED ARRAY [m..n:INTEGER ] OF CHAR;
                 rendition : typ_rendition );

{ IN : curwin : designateur d'une fenetre
       string : chaine de caracteres
       rendition : mode d'affichage video
  OUT : curwin modifie
  semantique : ajoute a l'ecran "curwin" la chaine de caracteres
               "string" ( en ignorant les 'trailing blanks'), a partir de
               la position courante de l'ecran.
               Si la chaine est trop longue, elle est tronquee. Le caractere
               dans le coin inferieur droit est force a blanc pour eviter
               un 'scrolling' lors d'un refresh ulterieur.
               Les caracteres de tabulation sont convertis en blancs.
}

VAR i,j : INTEGER;
    fini : BOOLEAN;
    longueur_utile,limite : INTEGER;

BEGIN
   longueur_utile := n - m + 1;
   i := n;
   WHILE i >= m DO
      BEGIN
         IF string[i] = ' ' THEN BEGIN
                                    i := i - 1;
                                    longueur_utile := longueur_utile - 1;
                                 END
                            ELSE i := m - 1;
  END;

IF cury(curwin) + longueur_utile <= maxy(curwin)
   THEN limite := longueur_utile
   ELSE IF curx(curwin) <> maxx(curwin)
           THEN limite := maxy(curwin) - cury(curwin) + m - 1
           ELSE limite := maxy(curwin) - cury(curwin) + m - 2;
                                  { un caractere affiche dans le coin
                                    inferieur droit causerait un scroll
                                    indesirable: on force donc ce
                                    caractere a blanc }

i := m;
WHILE (i <= limite) DO
   BEGIN
      IF string[i] <> CHR(9) THEN
         BEGIN
            addchar(curwin,string[i],rendition);
            i := i + 1 ;
         END
      ELSE
         BEGIN
            REPEAT
               addchar(curwin,' ',rendition);
            UNTIL ((cury(curwin) MOD 8) = 0) OR (cury(curwin) = maxy(curwin));
            i := i + 1;
         END;
```

```
      END {while};
END;

{-------------------------------------------------------------------------}

PROCEDURE box(VAR curwin : window; cx : CHAR; cy : CHAR);

{ IN : curwin : designateur d'une fenetre
       cx,cy : caracteres
  OUT : curwin modifiee
  semantique : trace un cadre sur les bords de la fenetre "curwin", en
               employant le caractere "cx" pour la dimension des x, et le
               caractere "cy" pour la dimension des y
}

VAR i : INTEGER;
    delta : INTEGER;   { 0 si la fenetre n'atteint pas le coin inferieur droit
                         1 si elle l'atteint : dans ce cas, ecrire dans le coin
                         entrainerait un scrolling indesirable, on laisse donc
                         les coins a blanc }
    debx,deby,finx,finy : INTEGER;

BEGIN
   debx := begx(curwin); deby := begy(curwin);
   finx := maxx(curwin); finy := maxy(curwin);
   IF finy = xphysy THEN delta := 1 ELSE delta := 0;

   move(curwin,debx,deby+delta);
   FOR i := deby+delta TO finy-delta DO
      addchar(curwin,cy,normal);
   move(curwin,finx,deby+delta);
   FOR i := deby+delta TO finy-delta DO
      addchar(curwin,cy,normal);


   FOR i := debx + 1 TO finx - 1 DO
      BEGIN
         move(curwin,i,deby);
         addchar(curwin,cx,normal);
         move(curwin,i,finy);
         addchar(curwin,cx,normal);
      END { for };
   touch(curwin);
END; {procedure box}

{-------------------------------------------------------------------------}

PROCEDURE ctrl_l;

{ OUT : action sur l'ecran du terminal.
  parametre global : curscr : designateur de la fenetre representant
                              l'aspect de l'ecran du terminal au moment
                              courant

  semantique : redessine l'ecran tel que le gestionnaire d'ecran le
               connait
}

BEGIN
   refresh(curscr);
```

```
{=========================================================================}

END;
```

```
PROCEDURE wipefield(VAR curwin : window;nb_car : INTEGER);

{ IN : curwin : designateur d'une fenetre
       nb_car : entier
  OUT : action sur l'ecran du terminal
  parametre global : ligne_blanche : chaine de caracteres, de longueur
                                     egale a la largeur maximale de
                                     l'ecran, garnie de blancs
  semantique : efface "nb_car" caracteres sur l'ecran du terminal,sur la
               ligne "curx" de "curwin",a partir de la position "cury",
                SANS AFFECTER L'ECRAN LOGIQUE.
}

BEGIN
   xmove(curx(curwin),cury(curwin));
   WRITE(TTY,ligne_blanche:nb_car);
END;

{-------------------------------------------------------------------------}

PROCEDURE wipe(VAR curwin : window);

{ IN : curwin : designateur d'une fenetre
  semantique : provoque un effacage de l'ecran du terminal lors du prochain
               'refresh'. Cet effacage a lieu sur l'etendue de la fenetre
               "curwin". Le contenu de l'ecran n'est pas affecte
}

BEGIN
   curwin^.clear := TRUE;
END;

{-------------------------------------------------------------------------}

PROCEDURE erase(VAR curwin : window);

{ IN : curwin : designateur d'une fenetre
  OUT : curwin modifiee
  semantique : provoque un effacage du contenu de la fenetre "curwin".
               L'ecran du terminal n'est pas affecte }

BEGIN
   init_fields(curwin,FALSE);
   touch(curwin);
END;


{-------------------------------------------------------------------------}

PROCEDURE clearall(VAR curwin : window);

{ IN : curwin : designateur d'une fenetre
  OUT : curwin modifiee
  semantique : provoque un effacage du contenu de la fenetre "curwin" et
               de l'ecran du terminal(effectif au prochain 'refresh')
}

BEGIN
```

```
    wipe(curwin);
    erase(curwin)
END;

{=============================================================================}
```

```
PROCEDURE overwrite(VAR win1,win2 : window);

{ window dependent }

{ IN : win1, win2 : designateur de fenetres
  OUT :  win2 modifiee
  semantique : reecrit le contenu de la fenetre "win1" sur la fenetre
               "win2" de facon destructive : les blancs sur "win1" deviennent
               des blancs sur "win2"
}

VAR i,j,debx,finx,deby,finy : INTEGER;

BEGIN
  WITH win2^ DO
     BEGIN
        IF win1^.begx > begx THEN debx := win1^.begx ELSE debx := begx;
        IF win1^.maxx < maxx THEN finx := win1^.maxx ELSE debx := maxx;
        IF win1^.begy > begy THEN deby := win1^.begy ELSE deby := begy;
        IF win1^.maxy < maxy THEN finy := win1^.maxy ELSE deby := maxy;
        FOR i:= debx TO finx DO
           BEGIN
              ar_mod[i] := TRUE;
              FOR j:= deby TO finy DO
                 BEGIN
                    content[i,j] := win1^.content[i,j];
                    rendition[i,j] := win1^.rendition[i,j];
                 END;
           END;
        END;
END;

{-------------------------------------------------------------------------}

PROCEDURE overlay(VAR win1,win2 : window);

{window dependent}

{ IN : win1, win2 : designateur de fenetres
  OUT :  win2 modifiee
  semantique : reecrit le contenu de la fenetre "win1" sur la fenetre
               "win2" de facon non-destructive : a l'emplacement des blancs
               de "win1", l'ancien contenu de "win2" subsiste
}

VAR i,j,debx,finx,deby,finy : INTEGER;

BEGIN
   WITH win2^ DO
      BEGIN
         IF win1^.begx > begx THEN debx := win1^.begx ELSE debx := begx;
         IF win1^.maxx < maxx THEN finx := win1^.maxx ELSE debx := maxx;
         IF win1^.begy > begy THEN deby := win1^.begy ELSE deby := begy;
         IF win1^.maxy < maxy THEN finy := win1^.maxy ELSE deby := maxy;
         FOR i:= begx TO maxx DO
            BEGIN
               ar_mod[i] := TRUE;
               FOR j:= begy TO maxy DO
```

```
                    IF win1^.content[i,j] <> ' ' THEN
                    BEGIN
                       content[i,j] := win1^.content[i,j];
                       rendition[i,j] := win1^.rendition[i,j];
                    END;
                END;
        END;
END;

{=============================================================================}
```

```
PROCEDURE save(curwin : window; nom_win : str10;
                        VAR nom_fich : PACKED ARRAY [m..n:INTEGER] OF CHAR;
                        VAR code_ret : typ_code_ret );

{ window dependent}

{ IN : curwin : designateur de fenetre
       nom_win,nom_fich : chaines de caracteres
  OUT : code_ret : code retour
  semantique : Sauve le contenu de la fenetre "curwin" , sous le nom
               "nom_win" dans le fichier nomme "nom_fich". Le code de
               retour "code_ret" vaut 'ok' si tout s'est bien passe, ou un
               code signalant l'anomalie rencontree sinon.

          Valeur des codes retour :
               ok
               wed : une fenetre ayant nom donne en argument existe deja

}

LABEL 100;

VAR fichier : f_window;
    c : CHAR;

BEGIN
   code_ret := ok;
   RESET(fichier,nom_fich,'/0');
                          {on essaie d'ouvrir le fichier en ne tenant
                           pas compte d'une eventuelle erreur a
                           l'ouverture (fichier non-existant)}
   IF erstat(fichier) <> 0 THEN
        REWRITE(fichier,nom_fich)
   ELSE
      BEGIN
        WHILE (NOT EOF(fichier)) AND (nom_win <> fichier^.nom_win) DO

              GET(fichier);

        IF NOT EOF(fichier) THEN
           BEGIN
              code_ret:=wed;
              GOTO 100;
           END {if};
      END {cas ou le fichier existe};

   append(fichier,nom_fich);

   fichier^ := curwin^;
   fichier^.nom_win := nom_win;
   PUT(fichier);
100:
END {procedure save};

{-------------------------------------------------------------------------}

PROCEDURE retrieve(VAR curwin : window;nom_win : str10;
                        VAR nom_fich : PACKED ARRAY [m..n:INTEGER] OF CHAR ;
```

```
                        VAR code_ret : typ_code_ret);

{curwin dependent}

{ IN : curwin : designateur d'une fenetre
       nom_win, nom_fich : chaines de caracteres
  OUT : code_ret : code retour
  semantique : Cree la fenetre "curwin" et la garnit avec le contenu de
               la fenetre enregistree sous le nom "nom_win" dans le fichier
               nomme "nom_fich".
               Le code de retour "code_ret" vaut 'ok' si tout s'est bien
               passe, ou un code signalant l'anomalie rencontree dans le
               cas contraire.
               Si une fenetre "curwin" existait deja, son contenu est
               perdu.

        Valeurs du code retour :
               ok
               fne : fichier (des fenetres) n'existe pas
               ept : ecran (a restaurer) pas trouve

}

LABEL 100;

VAR fichier : f_window;
    c,c_temp:CHAR;

BEGIN
   code_ret := ok;
   RESET(fichier,nom_fich,'/0');
   IF erstat(fichier) <> 0 THEN
     BEGIN
       code_ret:= fne;
       GOTO 100;
     END
   ELSE
      BEGIN
        WHILE (NOT EOF(fichier)) AND (nom_win <> fichier^.nom_win)  DO

              GET(fichier);

        IF EOF(fichier) THEN
           BEGIN
              code_ret := ept;
              GOTO 100;
           END {if};
      END {else};

   IF curwin <> NIL THEN DISPOSE(curwin);
   curwin := newwin(fichier^.begx,fichier^.begy,fichier^.maxx,fichier^.maxy);
   curwin^ := fichier^;
   touch(curwin);
100:
END {procedure retrieve};

{-------------------------------------------------------------------------}

PROCEDURE del_fich(VAR nom_fich : PACKED ARRAY [m..n:INTEGER] OF CHAR;
                   VAR code_ret : typ_code_ret);
```

```
{ IN : nom_fich : chaine de caracteres
  OUT : code_ret : code retour
  semantique : efface le fichier de nom "nom_fich".
               Le code de retour "code_ret" vaut 'ok' si tout s'est bien
               passe, ou un code signalant l'anomalie rencontree dans le
               cas contraire.

          Valeurs du code retour :
               ok
               fne : fichier (a effacer) n'existe pas

}

VAR fich : f_window;
    c : CHAR;

BEGIN
   code_ret := ok;
   RESET(fich,nom_fich,'/0');
   IF erstat(fich) = 0 THEN
      delete(fich)
   ELSE
      code_ret := fne;
END;

{-------------------------------------------------------------------------}

PROCEDURE del_win(nom_win : str10;
                  VAR nom_fich : PACKED ARRAY [m..n:INTEGER] OF CHAR ;
                  VAR code_ret : typ_code_ret);

{ IN : nom_win, nom_fich : chaines de caractere
  OUT : code_ret : code retour
  semantique : efface la fenetre de nom "nom_win" du fichier nomme
               "nom_fich". Le code de retour "code_ret" vaut 'ok' si tout
               s'est bien passe, ou un code signalant l'anomalie rencontree
               dans le cas contraire.

       Valeurs du code_retour :
           ok
           fne : fichier (dont on veut effacer une fenetre) n'existe pas
           wne : fenetre (que l'on veut effacer) n'existe pas
}

VAR fich,t_fich : f_window;
    trouve : BOOLEAN;
    c : CHAR;

BEGIN
   code_ret := ok;
   trouve := FALSE;
   RESET(fich,nom_fich,'/0');
   IF erstat(fich) <> 0
      THEN code_ret := fne
      ELSE
         BEGIN
            REWRITE(t_fich);
            WHILE (NOT EOF(fich)) DO
               BEGIN
```

```
                    IF (fich^.nom_win <> nom_win) THEN
                       BEGIN
                          t_fich^ := fich^;
                          PUT(t_fich);
                       END
                      ELSE trouve := TRUE;
                    GET(fich);
                 END; { while }
              IF NOT trouve THEN code_ret := wne
                            ELSE
                                rename(t_fich,nom_fich);

           END { if fichier existe }
END;

{=============================================================================}
```

```
    IF cadre THEN BEGIN
                    debx := 1;
                    finx := lim_lin;
                  END
             ELSE BEGIN
                    debx := begx(curwin);
                    finx := maxx(curwin);
                  END;
    FOR i := 1 TO lim_col DO
       BEGIN
          l_un[i] := n_to_s(i MOD 10);
          l_diz[i] := ' ';
       END;
    FOR i := 1 TO (lim_col DIV 10) DO l_diz[i*10] := n_to_s(i);
    IF cadre THEN
       BEGIN
          WRITELN(fimpr,'    ',l_diz);
          WRITELN(fimpr,'    ',l_un);
       END;
    FOR i := debx TO finx DO imp_ligne(i,cadre);
    IF cadre THEN
       BEGIN
          WRITELN(fimpr,'    ',l_un);
          WRITELN(fimpr,'    ',l_diz);
       END;
    page(fimpr);
END;

{=============================================================================}

PROCEDURE hard_copy(cadre : BOOLEAN);

{ IN : cadre : booleen
  parametre global : fichier d'impression 'spooler' de nom interne "fimpr"
                     curscr : designateur de la fenetre representant
                              l'aspect de l'ecran du terminal au moment
                              courant

  semantique : realise une copie sur le fichier d'impression de l'ecran
  physique du terminal tel que le connait le gestionnaire d'ecran}

BEGIN
   imp_ecran(curscr,cadre);
END;

{=============================================================================}
```

```
{ window dependent }

PROCEDURE readstr(x,y : INTEGER;
                  n_char_to_get : INTEGER;
                  VAR result : PACKED ARRAY [m..n:INTEGER] OF CHAR;
                  VAR count : INTEGER;
                  break_set : char_set;
                  rendition : typ_rendition;
                  VAR code_ret : typ_code_ret);


{ IN : x ,y  ; entiers
       break_set : ensemble de caracteres
       rendition : mode d'affichage video
  OUT : n_char_to_get, count : entiers
        result : chaine de caracteres
        code_ret : code_retour
  semantique : garnit le tableau de caracteres "result" avec des caracteres
               pris au terminal, a partir de la position courante de l'ecran
               "curwin", SANS MODIFIER CET ECRAN D'AUCUNE FACON, et s'arrete
               lorsque
                 - "n_char_to_get" caracteres ont ete lus
                 - un caractere repris dans "break_set" est rencontre
                 - le tableau "result" est rempli
              "count" contient le nombre de caracteres effectivement lus;
              En cours d'introduction, il est possible d'utiliser les touches
              "delete" et Control-U

  Des fonctions annexes sont fournies :
    - une facilite pour une aide eventuelle :
        si un '?' est introduit comme premier caractere, la lecture s'arrete
        et le code de retour est positionne pour indiquer cette eventualite.
    - si le caractere introduit est CNTRL-P :
        On obtient une copie de l'ecran courant a l'imprimante
    - si on introduit CNTRL-L :
        on provoque un rafraichissement de l'ecran (au cas ou il aurait ete
        perturbe par un message du systeme)

              Valeurs du code_retour :
                ok
                inter : le premier caractere lu est un "?"
}


VAR i,lim_tab : INTEGER;
    c : CHAR;
    break : BOOLEAN;


BEGIN
   xmove(x,y);
   code_ret := ok;
   break := FALSE;
   FOR i := m TO n DO result[i] := ' ';
   IF n_char_to_get > (n-m + 1) THEN n_char_to_get  := n-m +1;
   IF (y + n_char_to_get - 1) > xphysy THEN
      BEGIN
         n_char_to_get := xphysy - y + 1;
         IF x = xphysx THEN n_char_to_get := n_char_to_get - 1 ;
```

```
   END;
                                  { pour eviter d'ecrire dans le coin
                                    inferieur droit de l'ecran, ce qui
                                    causerait un scrolling indesirable
                                    pour une gestion d'ecran }

IF rendition <> normal THEN xso(rendition);
count := 0;
WHILE NOT break AND ( count < n_char_to_get ) DO
   BEGIN
      getc(c);
      IF c IN break_set THEN break := TRUE
      ELSE IF (c = '?') AND (count = 0) THEN BEGIN
                                               code_ret := inter;
                                               break := TRUE;
                                             END
        { CNTRL-P }
      ELSE IF c = CHR(16) THEN hard_copy(FALSE)
        { CNTRL-L }
      ELSE IF c = CHR(12) THEN BEGIN
                                 ctrl_l;
                                 xmove(x,y);
                                 WRITE(TTY,result:count);
                               END
        { DEL }
      ELSE IF c = CHR(127) THEN BEGIN IF count = 0  THEN bell
                                      ELSE
                                        BEGIN
                                          count := count - 1;
                                          xcb;
                                          WRITE(TTY,' ');
                                          xcb;
                                          result[m+count] := ' ';
                                        END
                                 END
  { CNTRL-U }
ELSE IF c = CHR(21) THEN BEGIN IF count = 0
                                  THEN bell
                                  ELSE
                                    REPEAT
                                      count := count - 1 ;
                                      xcb;
                                      WRITE(TTY,' ');
                                      xcb;
                                      result[m+count] := ' ';
                                    UNTIL count = 0 ;
                            END
  { TAB }
ELSE IF c = CHR(9) THEN BEGIN
                          lim_tab := ((((y+count)DIV 8)+1)*8);
                          REPEAT
                            WRITE(TTY,' ');
                            result[m+count] := ' ';
                            count := count + 1;
                          UNTIL ((y+count-1)=lim_tab) OR (count = n_char_to_get);
                        END
ELSE BEGIN
       result[m+count] := c;
       count := count + 1;
       WRITE(TTY,c);
```

```
                END;
        END ; { while }
    IF rendition <> normal THEN xse;
END { procedure readstr }.
```

```
PROGRAM screen;

include 'sys:string.dec';
include 'sys:cod_ret.dec';
include 'sys:utilit.dec';
include 'sys:tscr.dec';
include 'sys:getedi.dec';

LABEL 100;

VAR code_ret : typ_code_ret;
    choix,l_choix : str12;
    defaut_fich : str10;

{
FUNCTION s_to_n(VAR str : PACKED ARRAY [m..n:INTEGER] OF CHAR): INTEGER;
pour memoire; declare en include
}

{-------------------------------------------------------------------------}

PROCEDURE adjustxy(curwin : window;c : CHAR);

VAR x,y :INTEGER;

BEGIN
   x := curx(curwin);
   y := cury(curwin);
   CASE c OF
      '^F' : IF y < maxy(curwin) THEN y := y + 1 ELSE bell;
      '^B' : IF y > begy(curwin)  THEN y := y - 1 ELSE bell;
      '^N' : IF x < maxx(curwin) THEN x := x + 1 ELSE bell;
      '
```

```
    ' : IF x > begx(curwin)  THEN x := x - 1 ELSE bell
      END {case};
      move(curwin,x,y);
END {adjustxy};

{-------------------------------------------------------------------------------}

PROCEDURE gotoborder(curwin : window; c : CHAR);

VAR lin,col : INTEGER;

BEGIN
   lin := curx(curwin);
   col := cury(curwin);
   CASE c OF
      '^A' : col := begy(curwin);
      '^E' : col := maxy(curwin);
      '^U' : lin := begx(curwin);
      '^D' : lin := maxx(curwin);
   END { case };
   move(curwin,lin,col);
END { procedure gotoborder };

{-------------------------------------------------------------------------------}

PROCEDURE edition(VAR curwin : window);

VAR rendition : typ_rendition;
    string : PACKED ARRAY [1..lim_col] OF CHAR;
    c : char;
    long,i,l_max : INTEGER;
    code_ret : typ_code_ret;
    break : SET OF CHAR;

BEGIN
   move(curwin,begx(curwin),begy(curwin));
   break := [CHR(13)];
   xcls;
   noecho;
   refresh(curwin);
   getc(c);
   WHILE (c <> 'q') DO
      BEGIN
         CASE c OF
            '^N','^F','
```

```
',^B' : BEGIN
            adjustxy(curwin,c);
         END;
         ^A',^E',^U',^D' : BEGIN
            gotoborder(curwin,c);
         END;
         ^X' : BEGIN
            l_max := maxy(curwin) - cury(curwin) + 1;
            IF curx(curwin) = xphysx THEN l_max := l_max - 1;
            echo;
            aff_2msg(crend,chrend,c);
            CASE c OF
               'n' : rendition := normal;
               's' : rendition := under;
               'c' : rendition := blink;
               'g' : rendition := bold;
               'i' : rendition := reverse;
            END; {case}
            readstr(curx(curwin),cury(curwin),l_max,string,long,break,rendition,code_ret);
            addstr(curwin,string,rendition);
            noecho;
         END {'x'};
         '
```

```
' : BEGIN
            l_max := maxy(curwin) - cury(curwin) + 1;
            IF curx(curwin) = xphysx THEN l_max := l_max - 1;
            echo;
            readstr(curx(curwin),cury(curwin),l_max,string,long,break,normal,code_ret);
            FOR i := 1 TO long DO addchar(curwin,' ',normal);
            noecho;
         END;
         'e' : erase(curwin);
         'c' : clearall(curwin);
         'w' : wipe(curwin);
         '


' : ctrl_l;
         't' : BEGIN
            touch(curwin);
         END;
         'r' : BEGIN
            refresh(curwin);
```

```
            END;
            'B' : BEGIN
               box(curwin,'|','-');
               move(curwin,begx(curwin) + 1,begy(curwin) + 1);
            END;
            '^H' : BEGIN
               aff_2msg(ocadr,chcadr,c);
               IF c = 'c' THEN hard_copy(TRUE) ELSE hard_copy(FALSE);
            END;
            others : bell;
         END {case};
         refresh(curwin);
         getc(c);
         END {while};
      echo;
      xcls;
END; { procedure edition }

{-------------------------------------------------------------------------}

PROCEDURE trt_creation;
{ parame`tre global : defaut_fich }

VAR rep,l_rep : str3;
    lin,col,temp : INTEGER;
    orx,ory : INTEGER;
    c_orx,c_ory,c_lin,c_col : str2;
    nom_win : str10;
    nom_fich : str10;
    code_ret : typ_code_ret;
    curwin : window;

BEGIN
   WRITE(TTY,'Origine de l''ecran? x (Defaut: 1) : ');
   READLN(TTY);READ(TTY,c_orx);
   IF c_orx = '  ' THEN orx := 1
      ELSE
         BEGIN
            temp := s_to_n(c_orx);
            IF temp > xphysx THEN orx := xphysx
                             ELSE orx := temp;
         END;
  WRITE(TTY,'                    y (Defaut: 1) : ');
  READLN(TTY);READ(TTY,c_ory);
  IF c_ory = '  ' THEN ory := 1
     ELSE
        BEGIN
           temp := s_to_n(c_ory);
           IF temp > xphysy THEN ory := xphysy
                            ELSE ory := temp;
        END;

  WRITE(TTY,'dimensions de l''ecran? max x : (Defaut: ',xphysx:1,') : ');
  READLN(TTY);READ(TTY,c_lin);
  IF c_lin = '  ' THEN lin := xphysx
     ELSE
        BEGIN
           temp := s_to_n(c_lin);
           IF temp > xphysx THEN lin := xphysx
                            ELSE lin := temp;
```

```
      END;
WRITE(TTY,'                          max y : (Defaut: ',xphysy:1,') : ');
READLN(TTY);READ(TTY,c_col);
IF c_col = '  ' THEN col := xphysy
   ELSE
      BEGIN
         temp := s_to_n(c_col);
         IF temp > xphysy THEN col := xphysy
                          ELSE col := temp;
      END;

curwin:= newwin(orx,ory,lin,col);

edition(curwin);

REPEAT
   WRITELN(TTY,'Desirez-vous sauver la fenetre sur disque ? ');
   READLN(TTY);READ(TTY,rep);
   lower(rep,l_rep);
UNTIL match(l_rep,'oui') OR match(l_rep,'non');
IF match(l_rep,'oui') THEN
   REPEAT
      REPEAT
         WRITE(TTY,'Nom du fichier de sauvetage :');
         IF defaut_fich <> bl10 THEN WRITELN(TTY,' (defaut : ',defaut_fich,' )');
         READLN(TTY);READ(TTY,nom_fich);
         IF nom_fich <> bl10 THEN defaut_fich := nom_fich
                             ELSE nom_fich := defaut_fich;
      UNTIL nom_fich <> bl10;

      WRITE(TTY,'Nom de l''ecran :');
      READLN(TTY);READ(TTY,nom_win);

      save(curwin,nom_win,nom_fich,code_ret);
      IF code_ret = ok THEN ecr_msg(sef)
                       ELSE ecr_msg(code_ret);
   UNTIL code_ret = ok;

   clearall(curwin);
END;

{-------------------------------------------------------------------------------}

PROCEDURE trt_modif;

VAR rep,l_rep : str3;
    nom_win,nom_fich : str10;
    code_ret : typ_code_ret;
    curwin : window;

BEGIN
   curwin := newwin(1,1,xphysx,xphysy);
   REPEAT
      REPEAT
         WRITE(TTY,'Nom du fichier de sauvetage :');
         IF defaut_fich <> bl10 THEN WRITELN(TTY,' (defaut : ',defaut_fich,' )');
         READLN(TTY);READ(TTY,nom_fich);
         IF nom_fich <> bl10 THEN defaut_fich := nom_fich
                             ELSE nom_fich := defaut_fich;
      UNTIL nom_fich <> bl10;
```

```
      WRITE(TTY,'Nom de l''ecran :');
      READLN(TTY);READ(TTY,nom_win);
      retrieve(curwin,nom_win,nom_fich,code_ret);
      IF code_ret = ok THEN ecr_msg(ref)
                       ELSE ecr_msg(code_ret);
   UNTIL code_ret = ok;

   edition(curwin);

   REPEAT
      WRITELN(TTY,'Desirez vous sauver les modifications effectuees ?');
      READLN(TTY);READ(TTY,rep);
      lower(rep,l_rep);
   UNTIL match(l_rep,'oui') OR match(l_rep,'non');

   IF match(l_rep,'oui') THEN
      BEGIN
         del_win(nom_win,nom_fich,code_ret);
         IF code_ret <> ok THEN ecr_msg(callguru)
            ELSE
               BEGIN
                  save(curwin,nom_win,nom_fich,code_ret);
                  IF code_ret <> ok THEN ecr_msg(callguru);
               END;
      END;


END;

{-------------------------------------------------------------------------------}

PROCEDURE trt_eff;

VAR nom_win,nom_fich : str10;
    rep,l_rep : str3;
    code_ret : typ_code_ret;

BEGIN
   REPEAT
      REPEAT
         WRITELN(TTY,'Quel est le nom du fichier des fenetres ? ');
         READLN(TTY);READ(TTY,nom_fich);
      UNTIL nom_fich <> bl10;
      REPEAT
         WRITELN(TTY,'Desirez vous effacer TOUTES les fenetres du fichier, ou UNE ? ');
         READLN(TTY);READ(TTY,rep);
         lower(rep,l_rep);
      UNTIL match(l_rep,'une') OR match(l_rep,'toutes');
      IF match(l_rep,'une') THEN
         BEGIN
            REPEAT
               WRITELN(TTY,'Quel est le nom de la fenetre a effacer ? ');
               READLN(TTY);READ(TTY,nom_win);
            UNTIL nom_win = bl10;
            del_win(nom_win,nom_fich,code_ret);
            IF code_ret = ok THEN ecr_msg(wef)
                             ELSE ecr_msg(code_ret);
         END
      ELSE
```

```
        BEGIN
           del_fich(nom_fich,code_ret);
           IF code_ret = ok THEN ecr_msg(fef)
                            ELSE ecr_msg(code_ret);
        END;
   UNTIL code_ret = ok;
END;

{=============================================================================}



BEGIN
   defaut_fich := b110;
   initscr(code_ret);
   IF code_ret <> ok THEN BEGIN ecr_msg(code_ret); GOTO 100 END;
   WRITELN(TTY,'Que desirez-vous faire ?');
   WRITELN(TTY,'    CREATION d''une fenetre');
   WRITELN(TTY,'    MODIFICATION d''une fenetre');
   WRITELN(TTY,'    EFFACAGE d''une ou de toutes les fenetres');
   WRITELN(TTY,'    QUITTER');
   READLN(TTY);READ(TTY,choix);
   lower(choix,l_choix);
   WHILE NOT match(l_choix,'quitter') DO
      BEGIN
         IF match(l_choix,'creation') THEN trt_creation
         ELSE IF match(l_choix,'modification') THEN trt_modif
         ELSE IF match(l_choix,'effacage') THEN trt_eff;
         WRITELN(TTY,'Que desirez-vous faire ?');
         WRITELN(TTY,'    CREATION d''une fenetre');
         WRITELN(TTY,'    MODIFICATION d''une fenetre');
         WRITELN(TTY,'    EFFACAGE d''une ou de toutes les fenetres');
         WRITELN(TTY,'    QUITTER');
         READLN(TTY);READ(TTY,choix);
         lower(choix,l_choix);
      END;
   xcls;
   endscr;
100:
END.
```

TSCR

```
sm-}                                    (* This says this module has no main *)

* Library of External procedures to support a physical screen *)
* Michel E Debar - Feb 84 *)
* Some minor modifications by Charles Bokor - Feb/Mar 84 *)

*
lors que j'ecrivais cet ensemble de routines, je me suis heurte a
n ensemble de problemes inattendus. Le plus curieux est le fait que,
yant beaucoup de declarations "forward", correspondant a mon style
suel de programmation, j'ai eu la desagreable surprise de constater
ue le compilateur Pascal ne tenait pas compte de la plupart de mes
eclarations....

'ai egalement pu constater d'autres erreurs, une des plus belles etant
ommentee dans add_cstr.
)

nclude 'sys:cod_ret.dec';
      { ce fichier doit comprendre au minimum la declaration de "typ_cod_ret"
        comme etant ok      (les operations se sont bien passees)
                    tnd     (terminal non defini)
                    tinc    (terminal inconnu)
                    fecrep  (fichier des ecrans n'existe pas)}

ONST
riou_ = 101b;
orsp_ = 27b;                            (* mtopr: get line speed *)
elay_buffer_size = 100;                 (* An array of nulls for padding *)
CREEN_ID_SIZE = 15;

im_lin = 30;                            { c'est ici que sont definies les dimensions  }
im_col = 80;                            { physiques maximales des ecrans que peut }
                                        { traiter ce gestionnaire d'ecran.}
                                        { Ces constantes pouvant etre utilisees }
                                        { ailleurs, elle DOIVENT etre reprises dans }
                                        { TSCR.DEC }

YPE
                                        (* Dec 20 word *)
ord36 = SET OF 0..35;
                                        (* CSTRnn is a terminal control *)
                                        (* sequence. The TXT part (LEN chars) *)
                                        (* is sent to the terminal, followed *)
                                        (* by padding to the tune of DELAY *)
                                        (* milliseconds.  *)

str7 = RECORD
  len:INTEGER;
  txt:PACKED ARRAY [1..7] OF CHAR;
  delay: INTEGER;
  END;
                                        (* CSEQnn is a terminal control *)
                                        (* sequence. Characters are *)
                                        (* represented by their ascii control *)
                                        (* code. Padding is represented by *)
                                        (* values > 127 as (val-127) *)
                                        (* milliseconds of padding. Sequence *)
                                        (* ends on a null.*)
seq30 = ARRAY [1..30] OF INTEGER;
```

```
                                        (* Type of cursor positioning algorithm *)
yp_curposalg = (vt52,vt100);
creen_id = PACKED ARRAY [1..SCREEN_ID_SIZE] OF CHAR;
hysical_screen = RECORD
  id1 : screen_id;
  id2 : screen_id;
  id3 : screen_id;
  height : INTEGER;
  width : INTEGER;
  init_screen: cseq30;
  reset_screen: cseq30;
  home: cstr7;
  clear_screen: cstr7;
  clear_eol: cstr7;
  clear_eos: cstr7;
  curpos_alg : typ_curposalg;
  curpos_line_first : BOOLEAN;
  curpos_line_offset : INTEGER;
  curpos_col_offset : INTEGER;
  curpos_lead : cstr7;
  curpos_middle : cstr7;
  curpos_end : cstr7;
  curpos_delay : INTEGER;
  start_reverse : cstr7;
  start_blink : cstr7;
  start_under : cstr7;
  start_bold : cstr7;
  end_highlight: cstr7;
  cur_up : cstr7;
  cur_do : cstr7;
  cur_fo : cstr7;
  cur_ba : cstr7;
ND;

ype_rendition = (normal,bold,blink,reverse,under);
                                { ce type etant utilise ailleurs, il DOIT }
                                { etre repris dans 'TSCR.DEC' }


AR
creen_file : FILE OF physical_screen;
c : physical_screen;
ine_speed: INTEGER;
uf80: PACKED ARRAY [1..80] OF CHAR;
elay_buffer: PACKED ARRAY [1..delay_buffer_size] OF CHAR;
nt_dum: INTEGER;
ld_tty_mode : word36;           (* TTY RFMOD mode word *)
ot_recu : word36;

ROCEDURE quit; EXTERN;
```

```
* err - print err message and die *)
ROCEDURE err (VAR msg : PACKED ARRAY [lo..hi:INTEGER] OF CHAR);

EGIN
  WRITELN (TTY, msg);
  quit;
ND;
```

```
(* ------- SUPPORT ROUTINES ------ *)
```

```
ROCEDURE snddel (del : INTEGER);

AR
nt: INTEGER;

EGIN
  cnt := del * line_speed DIV 1000;
  WHILE (cnt > 0) DO BEGIN
     IF (cnt > delay_buffer_size) THEN BEGIN
        jsys (53b; priou_, delay_buffer, - delay_buffer_size);
        cnt := cnt - delay_buffer_size;
     END (* if (cnt > delay_buffer_size) *)
     ELSE BEGIN
        jsys (53b; priou_, delay_buffer, - cnt);
        cnt := 0;
     END;
  END; (* while (cnt > 0) *)
ND;
```

```
(* send a char sequence, with eventual delay *)
'ROCEDURE sndcst (VAR cstr: PACKED ARRAY [lo..hi:INTEGER] OF CHAR;
                      len: INTEGER;
                      del:INTEGER);

'AR
'nt : INTEGER;

;EGIN
                                    (* send sequence *)
   IF (len > 0) THEN
      jsys (53b; priou_, cstr, - len);
                                    (* send delay if requested *)
   IF (del > 0) THEN
      snddel (del);
;ND;
```

```
 * add a numeric parameter to a buffer *)
ROCEDURE addprm (par:INTEGER;
                 VAR buf: PACKED ARRAY [lo..hi:INTEGER] OF CHAR;
                 VAR pbuf:INTEGER);

AR
 : INTEGER;

EGIN
  IF (par > 999) THEN
    err ('Can''t addprm greater than 999.');
  p := par;
  IF (p > 99) THEN BEGIN
    pbuf := pbuf+1;
    buf[pbuf] := CHR(48 + (p DIV 100) );
    p := p MOD 100;
  END; (* if (p > 99) *)

  IF (p > 9) THEN BEGIN
    pbuf := pbuf+1;
    buf[pbuf] := CHR(48 + (p DIV 10) );
    p := p MOD 10;
  END; (* if (p > 9) *)

  pbuf := pbuf+1;
  buf[pbuf] := CHR(48 + p);

ND;
```

```
(* add a cstr a bufffer *)
(* On entry to the procedure pbuf must point to the next unused entry
   in the buffer *)

PROCEDURE addcst (VAR cstr: PACKED ARRAY [cstr_lo..cstr_hi:INTEGER] OF CHAR;
                  len: INTEGER;
                  VAR buf: PACKED ARRAY [buf_lo..buf_hi:INTEGER] OF CHAR;
                  VAR pbuf: INTEGER);

VAR
  i : INTEGER;
  bug: BOOLEAN;

(*
A nice bug:
In the code below, if one writes:
   if (len > cstr_hi) or (pbuf+len > buf_hi) then
      err ('Bounds exceeded in addcst.');
One gets the message that it is not allowed to set the bounds
of a conformant array, or the control variable of a FOR loop...
 .... "bizarre, vous avez dit 'bizarre' ..."
*)

BEGIN
   bug := FALSE;
   IF (len > cstr_hi) OR (pbuf+len > buf_hi) THEN
      bug := TRUE;
   IF bug THEN
      err ('Bounds exceeded in addcst.');

   FOR i := 1 TO len DO BEGIN
      pbuf := pbuf + 1;
      buf[pbuf] := cstr[i]
   END; (* for i := 1 to len *)
END;
```

```
(* procedure to put in buffer code from a cseq *)

PROCEDURE sndcsq (VAR seq: ARRAY [seq_lo..seq_hi:INTEGER] OF INTEGER);

LABEL
100;

CONST
BUF_SIZE = 500;

VAR
i:INTEGER;
cnt:INTEGER;
pbuf:INTEGER;
buf: PACKED ARRAY [1..BUF_SIZE] OF CHAR;

BEGIN
   pbuf := 0;
   FOR i := 1 TO buf_size DO buf[i] := CHR(0);
   FOR i := seq_lo TO seq_hi DO BEGIN
      IF (seq[i] = 0) THEN GOTO 100;
      IF (seq[i] < 128) THEN BEGIN
         pbuf := pbuf+1;
         buf[pbuf] := CHR(seq[i]);
      END (* if (seq[i] < 128) *)
      ELSE BEGIN
         cnt := (seq[i]-127) * line_speed DIV 1000;
         WHILE (cnt > 0) DO BEGIN
            pbuf := pbuf + 1;
            buf[pbuf] := CHR(0);
            cnt := cnt - 1;
         END; (* while (cnt > 0) *)
      END;
   END; (* for i := 1 to top_seq *)
100:;
   jsys (53b; priou_, buf, - pbuf);

END;

{ fin des procedures de support }
```

```
{ Les procedures qui suivent sont celles qui doivent etre accessible de
  l'exterieur; leurs declarations doivent donc etre reprises dans 'TSCR.DEC'}

(* Init screen *)

PROCEDURE xiscr(VAR code_ret : typ_code_ret);

LABEL 100;

VAR
i: INTEGER;
sc_id : screen_id;
js_ret : INTEGER;
found : BOOLEAN;

BEGIN
code_ret := ok;
                                (* Initialize some stuff *)
FOR i := 1 TO delay_buffer_size DO
   delay_buffer[i] := CHR(0);

                                (* Get the type of screen *)
sc_id :=                    ';
jsys (504b,2,js_ret; 0, 'term', sc_id);
                                (* Replace nulls by space *)
   FOR i := 1 TO SCREEN_ID_SIZE DO
      IF ORD(sc_id[i]) = 0 THEN
         sc_id[i] := CHR(32);

IF js_ret = 1 THEN BEGIN
   code_ret:=tnd;
   GOTO 100;
END;
                                (* Look up description *)
RESET (screen_file, 'SYS:SCREENS.DESC','/O');
IF EOF(screen_file) THEN BEGIN
                           code_ret := fecrep;
                           GOTO 100;
                         END;
found := FALSE;
WHILE (NOT EOF(screen_file)) AND (NOT found) DO BEGIN
   READ (screen_file, sc);
   found := (sc_id = sc.id1) OR (sc_id = sc.id2) OR (sc_id = sc.id3);
END; (* while (not eof(screen_file)) and (not found) *)
close (screen_file);
IF (NOT found) THEN BEGIN
   code_ret := tinc;
   GOTO 100;
END; (* if (not found) *)

                                (* Get Line Speed (needed for padding) *)
                                (* Line speed is in chars / second *)
jsys (77b; priou_, morsp_; int_dum, int_dum, line_speed);
line_speed := (line_speed MOD 262144) DIV 10;

                                (* Set terminal in 8 bit mode *)
                                (* get mode word *)
jsys (107b; priou_; int_dum , old_tty_mode);
```

```
                                    (* turn off output control *)
isys (110b; priou_, (old_tty_mode - [28,29]));

                                    (* Send Init sequence *)
sndcsq (sc.init_screen);
100:
END;
```

```
(* REset Screen *)
PROCEDURE xrscr;
BEGIN
   sndcsq (sc.reset_screen);
                                (* Reset tty to orignl mode *)
jsys (110b; priou_, old_tty_mode);
END;
```

```
(* clear screen *)
PROCEDURE xcls;
BEGIN
WITH sc DO BEGIN
   sndcst (home.txt,home.len,home.delay);
   sndcst (clear_eos.txt,clear_eos.len,clear_eos.delay);
END;
                                (* Reset terminal position *)
jsys(526b; priou_, 0:0 );
END;

(* xcltoe *)
PROCEDURE xcltoe;
BEGIN
WITH sc DO
   sndcst (clear_eos.txt,clear_eos.len,clear_eos.delay);
END;

(* xclli *)
PROCEDURE xclli;
BEGIN
WITH sc DO
   sndcst (clear_eol.txt,clear_eol.len,clear_eol.delay);
END;

(* xso *)
PROCEDURE xso(rendition : typ_rendition);
BEGIN
WITH sc DO
   BEGIN
      CASE rendition OF
         reverse : sndcst (start_reverse.txt,start_reverse.len,start_reverse.delay);
         blink : sndcst (start_blink.txt,start_blink.len,start_blink.delay);
         under : sndcst (start_under.txt,start_under.len,start_under.delay);
         bold : sndcst (start_bold.txt,start_bold.len,start_bold.delay);
      END;
   END;
END;

(* xse *)
PROCEDURE xse;
BEGIN
WITH sc DO
   sndcst (end_highlight.txt,end_highlight.len,end_highlight.delay);
END;

{PROCEDURE xsoscr (VAR lin,col : INTEGER);
BEGIN
WITH sc DO
   BEGIN
      lin := height;
      col := width
   END
END;}

FUNCTION xphysx : INTEGER;
BEGIN
  xphysx := sc.height;
END;
```

```
FUNCTION xphysy : INTEGER;
BEGIN
  xphysy := sc.width;
END;

PROCEDURE xcu;
VAR line,col : INTEGER;
BEGIN
WITH sc DO
   sndcst(cur_up.txt,cur_up.len,cur_up.delay);
   jsys(111b;priou_,line:col);
   jsys(526b;priou_,(line-1):col);
END;

PROCEDURE xcd;
VAR line,col : INTEGER;
BEGIN
WITH sc DO
   sndcst(cur_do.txt,cur_do.len,cur_do.delay);
   jsys(111b;priou_,line:col);
   jsys(526b;priou_,(line+1):col);
END;

PROCEDURE xcf;
VAR line,col : INTEGER;
BEGIN
WITH sc DO
   sndcst(cur_fo.txt,cur_fo.len,cur_fo.delay);
   jsys(111b;priou_,line:col);
   jsys(526b;priou_,line:(col+1));
END;

PROCEDURE xcb;
VAR line,col : INTEGER;
BEGIN
WITH sc DO
   sndcst(cur_ba.txt,cur_ba.len,cur_ba.delay);
   jsys(111b;priou_,line:col);
   jsys(526b;priou_,line:(col-1));
END;


PROCEDURE echo;

VAR modificateur,mot_modif,bidon : word36;

BEGIN
   modificateur := [24];
   jsys(107b;priou_;bidon,mot_recu);
   mot_modif := mot_recu + modificateur;
   jsys(110b;priou_,mot_modif);
END;


PROCEDURE noecho;

VAR modificateur,mot_modif,bidon : word36;

BEGIN
```

```
    modificateur := [24];
    jsys(107b;priou_;bidon,mot_recu);
    mot_modif := mot_recu - modificateur;
    jsys(110b;priou_,mot_modif);
END;
```

```
(*
XMOVE - Move physical cursor somewhere on the screen.
Line and column positions are accepted starting from 1,1 beeing at
the top left corner.
If any of the two exceeds the size of the physical screen, we DO NOTHING.
*)

PROCEDURE xmove (line, col : INTEGER);

LABEL
100;

VAR
buf: PACKED ARRAY [1..80] OF CHAR;
pbuf : INTEGER;

BEGIN

                                (* BUILD ADDRESSING SEQ IN BUF *)
   WITH sc DO BEGIN
                                (* check limits *)
      IF (line > height) OR (col > width) THEN
         GOTO 100;

      pbuf := 0;
                                (* VT52 like addressing *)
      IF (curpos_alg = vt52) THEN BEGIN
         addcst (curpos_lead.txt,curpos_lead.len, buf80, pbuf);
         pbuf := pbuf + 1;
         IF (curpos_line_first) THEN
            buf80[pbuf] := CHR(line-1+curpos_line_offset)
         ELSE
            buf80[pbuf] := CHR(col-1+curpos_col_offset);
         pbuf := pbuf + 1;
         IF (curpos_line_first) THEN
            buf80[pbuf] := CHR(col-1+curpos_col_offset)
         ELSE
            buf80[pbuf] := CHR(line-1+curpos_line_offset);
      END
                                (* VT100 like addressing *)
      ELSE IF (curpos_alg = vt100) THEN BEGIN
         addcst (curpos_lead.txt,curpos_lead.len, buf80, pbuf);
         addprm (line, buf80, pbuf);
         addcst (curpos_middle.txt,curpos_middle.len, buf80, pbuf);
         addprm (col, buf80, pbuf);
         addcst (curpos_end.txt, curpos_end.len, buf80, pbuf);
      END;
                                (* SEND ADDRESSING SEQ *)
      isys (53b; priou_, buf80, -pbuf);
                                (* SEND PADDING IF NEEDED *)
      IF (curpos_delay > 0) THEN
         snddel (curpos_delay);
                                (* Reset terminal position *)
      isys(526b; priou_, (line-1):(col-1));
100:;
   END; (* with sc *)
END;
```

```
PROCEDURE xclear(debx,deby,finx,finy : INTEGER);

{ effacage de l'ecran physique, mais n'agissant que sur l'etendue de la
  fenetre fournie en argument }

VAR i,j : INTEGER;
    ligne_blanche : PACKED ARRAY [1..lim_col] OF CHAR;

BEGIN
   IF (debx = 1) AND (deby = 1) AND (finx = lim_lin)
                  AND (finy = lim_col) THEN xcls
   ELSE IF (finx = lim_lin) AND (finy = lim_col) AND (deby = 1) THEN
                BEGIN
                   xmove(debx,deby);
                   xcltoe
                END
   ELSE IF (deby = 1) AND (finy = lim_col) THEN
                              FOR i := debx TO finx DO
                                    BEGIN
                                       xmove(i,1);
                                       xclli
                                    END
   ELSE IF (finy = lim_col) THEN
                              BEGIN
                                 FOR i := debx TO finx DO
                                    BEGIN
                                       xmove(i,deby);
                                       xclli
                                    END {for};
                                 xmove(debx,deby);
                              END
   ELSE BEGIN
          FOR i:=1 TO lim_col DO ligne_blanche[i] := ' ';
          i := finy - deby + 1 ;
          FOR i := debx TO finx DO
             BEGIN
                xmove(i,deby);
                WRITE(TTY,ligne_blanche:i);
             END;
          xmove(debx,deby);
        END;
END {procedure xclear}.
```

MKDESC

```
(* Prepare description of physical screens *)
(* Michel E Debar - Feb 84 *)

(*
Alors que j'ecrivais cet ensemble de routines, je me suis heurte a
un ensemble de problemes inattendus. Le plus curieux est le fait que,
ayant beaucoup de declarations "forward", correspondant a mon style
usuel de programmation, j'ai eu la desagreable surprise de constater
que le compilateur Pascal ne tenait pas compte de la plupart de mes
declarations....

J'ai egalement pu constater d'autres erreurs, une des plus belles etant
commentee dans add_cstr.
*)

CONST
PRIOU_ = 101b;
MORSP_ = 27b;                   (* mtopr: get line speed *)
DELAY_BUFFER_SIZE = 100;        (* An array of nulls for padding *)
SCREEN_ID_SIZE = 15;


TYPE
                                (* CSTRnn is a terminal control *)
                                (* sequence. The TXT part (LEN chars) *)
                                (* is sent to the terminal, followed *)
                                (* by padding to the tune of DELAY *)
                                (* milliseconds.  *)

cstr7 = RECORD
   len:INTEGER;
   txt:PACKED ARRAY [1..7] OF CHAR;
   delay: INTEGER;
   END;
                                (* CSEQnn is a terminal control *)
                                (* sequence. Characters are *)
                                (* represented by their ascii control *)
                                (* code. Padding is represented by *)
                                (* values > 127 as (val-127) *)
                                (* milliseconds of padding. Sequence *)
                                (* ends on a null.*)
cseq30 = ARRAY [1..30] OF INTEGER;
                                (* Type of cursor positioning algorithm *)
typ_curposalg = (vt52,vt100);
screen_id = PACKED ARRAY [1..SCREEN_ID_SIZE] OF CHAR;
physical_screen = RECORD
   id1 : screen_id;
   id2 : screen_id;
   id3 : screen_id;
   height : INTEGER;
   width : INTEGER;
   init_screen: cseq30;
   reset_screen: cseq30;
   home: cstr7;
   clear_screen: cstr7;
   clear_eol: cstr7;
   clear_eos: cstr7;
   curpos_alg : typ_curposalg;
   curpos_line_first : boolean;
   curpos_line_offset : INTEGER;
```

```
    curpos_col_offset : INTEGER;
    curpos_lead : cstr7;
    curpos_middle : cstr7;
    curpos_end : cstr7;
    curpos_delay : INTEGER;
    start_reverse : cstr7;
    start_blink : cstr7;
    start_under : cstr7;
    start_bold : cstr7;
    end_highlight: cstr7;
    cur_up : cstr7;
    cur_do : cstr7;
    cur_fo : cstr7;
    cur_ba : cstr7;
END;

VAR
sf : FILE OF physical_screen;
sc : physical_screen;
line_speed: INTEGER;
buf80: PACKED ARRAY [1..80] OF CHAR;
delay_buffer: PACKED ARRAY [1..DELAY_BUFFER_SIZE] OF CHAR;
int_dum: INTEGER;


BEGIN

                              (* Open new description file *)
REWRITE (sf, 'SCREENS.DESC');
```

```
(* START SKELETON *)

WITH sc DO BEGIN
                                (* screen id, 15 chars, UPPERCASE *)
   id1 := '               ';
   id2 := '               ';
   id3 := '               ';
                                (* height, width *)
   height := 24;
   width := 80;
                                (* Screen initialization - CSEQ 30 *)
   init_screen[1] := 0;
                                (* Screen reset - CSEQ 30 *)
   reset_screen[1] := 0;
                                (* Put cursor home *)
   WITH home DO BEGIN
      len := 0;
      txt := '       ';
      delay :=0;
   END;
                                (* Clear screen *)
   WITH clear_screen DO BEGIN
      len := 0;
      txt := '       ';
      delay :=0;
   END;
                                (* Clear to end of line *)
   WITH clear_eol DO BEGIN
      len := 0;
      txt := '       ';
      delay :=0;
   END;

   WITH clear_eos DO BEGIN
      len := 0;
      txt := '       ';
      delay :=0;
   END;
                                (* Type of cursor positioning *)
                                (* VT52, VT100 *)
   curpos_alg := VT52;
                                (* True if line pos sent first *)
   curpos_line_first := TRUE;
                                (* Offset for line, column *)
   curpos_line_offset := 0;
   curpos_col_offset :=0;
                                (* Beginning string of cursor positionning *)
   WITH curpos_lead DO BEGIN
      len := 0;
      txt := '       ';
      delay :=0;
   END;
                                (* Middle string of cursor positionning *)
   WITH curpos_middle DO BEGIN
      len := 0;
      txt := '       ';
      delay :=0;
   END;
                                (* Final string of cursor positionning *)
```

```
WITH curpos_end DO BEGIN
   len := 0;
   txt := '        ';
   delay :=0;
END;
                                (* Delay aftr cursor positionning, millisecs *)
curpos_delay := 0;
                                (* Start reverse *)
WITH start_reverse DO BEGIN
   len := 0;
   txt := '        ';
   delay :=0;
END;
                                (* Start blink *)
WITH start_blink DO BEGIN
   len := 0;
   txt := '        ';
   delay :=0;
END;
                                (* Start under *)
WITH start_under DO BEGIN
   len := 0;
   txt := '        ';
   delay :=0;
END;
                                (* Start bold *)
WITH start_bold DO BEGIN
   len := 0;
   txt := '        ';
   delay :=0;
END;
                                (* End high visibility *)
WITH end_highlight DO BEGIN
   len := 0;
   txt := '        ';
   delay :=0;
END;
                                (* Cursor up *)
WITH cur_up DO BEGIN
   len := 0;
   txt :='        ';
   delay := 0;
END;
                                (* Cursor down *)
WITH cur_do DO BEGIN
   len := 0;
   txt :='        ';
   delay := 0;
END;
                                (* Cursor forward *)
WITH cur_fo DO BEGIN
   len := 0;
   txt :='        ';
   delay := 0;
END;
                                (* Cursor backward *)
WITH cur_ba DO BEGIN
   len := 0;
   txt :='        ';
   delay := 0;
```

```
    END;

END; (* with sc *)
                                    (* Now, write this term def out *)
(*
WRITE (sf, sc);
WRITELN (TTY, 'Stored definition for ', sc.id1,
        ',', sc.id2, ',', sc.id3, '.');
*)

(* END SKELETON *)
```

```
(* START --VT100 *)

WITH sc DO BEGIN
                                    (* screen id, 15 chars, UPPERCASE *)
   id1 := 'VT100           ';
   id2 := '                ';
   id3 := '                ';
                                    (* height, width *)
   height := 24;
   width := 80;
                                    (* Screen initialization - CSEQ 30 *)
   init_screen[1] := 0;
                                    (* Screen reset - CSEQ 30 *)
   reset_screen[1] := 0;
                                    (* Put cursor home *)
   WITH home DO BEGIN
      len := 3;
      txt := 'e[H    ';
      txt[1] := CHR(27);
      delay :=0;
   END;
                                    (* Clear screen *)
   WITH clear_screen DO BEGIN
      len := 6;
      txt := 'e[He[J ';
      txt[1] := CHR(27);
      txt[4]:= CHR(27);
      delay :=0;
   END;
                                    (* Clear to end of line *)
WITH clear_eol DO BEGIN
   len := 3;
   txt := 'e[K    ';
   TXT[1] := CHR(27);
   delay :=0;
END;

WITH clear_eos DO BEGIN
   len := 3;
   txt := 'e[J    ';
   txt[1] := CHR(27);
   delay :=0;
END;
                                    (* Type of cursor positioning *)
                                    (* VT52, VT100 *)
curpos_alg := VT100;
                                    (* True if line pos sent first *)
curpos_line_first := TRUE;
                                    (* Offset for line, column *)
curpos_line_offset := 0;
curpos_col_offset :=0;
                                    (* Beginning string of cursor positionning *)
WITH curpos_lead DO BEGIN
   len := 2;
   txt := 'e[     ';
   txt[1] := CHR(27);
   delay :=0;
END;
                                    (* Middle string of cursor positionning *)
```

```
WITH curpos_middle DO BEGIN
   len := 1;
   txt := '      ';
   delay :=0;
END;
                              (* Final string of cursor positionning *)
WITH curpos_end DO BEGIN
   len := 1;
   txt := 'H     ';
   delay :=0;
END;
                              (* Delay aftr cursor positionning, millisecs *)
curpos_delay := 0;
                              (* Start reverse *)
WITH start_reverse DO BEGIN
   len := 4;
   txt := '$[7m   ';
   txt[1] := CHR(27);
   delay :=0;
END;
                              (* Start blink *)
WITH start_blink DO BEGIN
   len := 0;
   txt := '       ';
   delay :=0;
END;
                              (* Start under *)
WITH start_under DO BEGIN
   len := 0;
   txt := '       ';
   delay :=0;
END;
                              (* Start bold *)
WITH start_bold DO BEGIN
   len := 0;
   txt := '       ';
   delay :=0;
END;
                              (* End high visibility *)
WITH end_highlight DO BEGIN
   len := 3;
   txt := 's[m    ';
   txt[1] := CHR(27);
   delay :=0;
END;
                              (* Cursor up *)
WITH cur_up DO BEGIN
   len := 2;
   txt :='$A     ';
   txt[1] := CHR(27);
   delay := 0;
END;
                              (* Cursor down *)
WITH cur_do DO BEGIN
   len := 2;
   txt :='$B     ';
   txt[1] := CHR(27);
   delay := 0;
END;
                              (* Cursor forward *)
```

```
    WITH cur_fo DO BEGIN
       len := 2;
       txt :='$C      ';
       txt[1] := CHR(27);
       delay := 0;
    END;
                                    (* Cursor backward *)
    WITH cur_ba DO BEGIN
       len := 2;
       txt :='$D      ';
       txt[1] := CHR(27);
       delay := 0;
    END;
END; (* WITH sc *)
                                    (* Now, write this term def out *)
WRITE (sf, sc);
WRITELN (TTY, 'Stored definition for ', sc.id1,
       ',', sc.id2, ',', sc.id3, '.');


(* END VT100 *)
```

```
(* START --AAA-- *)

WITH sc DO BEGIN
                                (* screen id, 15 chars, UPPERCASE *)
   id1 := 'AMBASSADOR     ';
   id2 := 'AAA            ';
   id3 := '               ';
                                (* height, width *)
   height := 30;
   width := 80;
                                (* Screen initialization - CSEQ 30 *)
   init_screen[1] := 27;        (* esc *)
   init_screen[2] := 91;        (* [ *)
   init_screen[3] := 54;        (* 60 *)
   init_screen[4] := 48;
   init_screen[5] := 59;        (* ;;; *)
   init_screen[6] := 59;
   init_screen[7] := 59;
   init_screen[8] := 51;        (* 30 *)
   init_screen[9] := 48;
   init_screen[10] := 112;      (* p *)
   init_screen[11] := 0;

                                (* Screen reset - CSEQ 30 *)
   reset_screen[1] := 0;
                                (* Put cursor home *)
   WITH home DO BEGIN
      len := 3;
      txt := '$[H    ';
      txt[1] := CHR(27);
      delay :=0;
END;
                                (* Clear screen *)
WITH clear_screen DO BEGIN
   len := 6;
   txt := '$[H$[J ';
   txt[1] := CHR(27);
   txt[4]:= CHR(27);
   delay :=0;
END;
                                (* Clear to end of line *)
WITH clear_eol DO BEGIN
   len := 3;
   txt := '$[K    ';
   TXT[1] := CHR(27);
   delay :=0;
END;

WITH clear_eos DO BEGIN
   len := 3;
   txt := '$[J    ';
   txt[1] := CHR(27);
   delay :=0;
END;
                                (* Type of cursor positioning *)
                                (* VT52, VT100 *)
curpos_alg := VT100;
                                (* True if line pos sent first *)
curpos_line_first := TRUE;
```

```
                                (* Offset for line, column *)
curpos_line_offset := 0;
curpos_col_offset :=0;
                                (* Beginning string of cursor positionning *)
WITH curpos_lead DO BEGIN
   len := 2;
   txt := '$[      ';
   txt[1] := CHR(27);
   delay :=0;
END;
                                (* Middle string of cursor positionning *)
WITH curpos_middle DO BEGIN
   len := 1;
   txt := ';       ';
   delay :=0;
END;
                                (* Final string of cursor positionning *)
WITH curpos_end DO BEGIN
   len := 1;
   txt := 'H       ';
   delay :=0;
END;
                                (* Delay aftr cursor positionning, millisecs *)
curpos_delay := 0;
                                (* Start reverse *)
WITH start_reverse DO BEGIN
   len := 4;
   txt := '$[7m    ';
   txt[1] := CHR(27);
   delay :=0;
END;
                                (* Start blink *)
WITH start_blink DO BEGIN
   len := 4;
   txt := '$[5m    ';
   txt[1] := CHR(27);
   delay :=0;
END;
                                (* Start under *)
WITH start_under DO BEGIN
   len := 4;
   txt := '$[4m    ';
   txt[1] := CHR(27);
   delay :=0;
END;
                                (* Start bold *)
WITH start_bold DO BEGIN
   len := 4;
   txt := '$[1m    ';
   txt[1] := CHR(27);
   delay :=0;
END;
                                (* End high visibility *)
WITH end_highlight DO BEGIN
   len := 3;
   txt := '$[m     ';
   txt[1] := CHR(27);
   delay :=0;
END;
                                (* Cursor up *)
```

```
    WITH cur_up DO BEGIN
       len := 3;
       txt :='$[A     ';
       txt[1] := CHR(27);
       delay := 0;
    END;
                                    (* Cursor down *)
    WITH cur_do DO BEGIN
       len := 3;
       txt :='$[B     ';
       txt[1] := CHR(27);
       delay := 0;
    END;
                                    (* Cursor forward *)
    WITH cur_fo DO BEGIN
       len := 3;
       txt :='$[C     ';
       txt[1] := CHR(27);
       delay := 0;
    END;
                                    (* Cursor backward *)
    WITH cur_ba DO BEGIN
       len := 3;
       txt :='$[D     ';
       txt[1] := CHR(27);
       delay := 0;
    END;
  END; (* with sc *)
                                    (* Now, write this term def out *)
  WRITE (sf, sc);
  WRITELN (TTY, 'Stored definition for ', sc.id1,
          ',', sc.id2, ',', sc.id3, '.');


  (* END AAA *)
```

```
(* START --GIGI-- *)

WITH sc DO BEGIN
                                (* screen id, 15 chars, UPPERCASE *)
   id1 := 'GIGI           ';
   id2 := '               ';
   id3 := '               ';
                                (* height, width *)
   height := 24;
   width := 80;

                                (* Screen initialization - CSEQ 30 *)
   init_screen[1] := 0;

                                (* Screen reset - CSEQ 30 *)
   reset_screen[1] := 0;

                                (* Put cursor home *)
   WITH home DO BEGIN
      len := 4;
      txt := '$[;H   ';
      txt[1] := CHR(27);
      delay :=0;
   END;

                                (* Clear screen *)
   WITH clear_screen DO BEGIN
      len := 4;
      txt := '$[2J   ';
      txt[1] := CHR(27);
      delay :=0;
   END;

                                (* Clear to end of line *)
WITH clear_eol DO BEGIN
   len := 3;
   txt := '$[K    ';
   TXT[1] := CHR(27);
   delay :=0;
END;

WITH clear_eos DO BEGIN
   len := 3;
   txt := '$[J    ';
   txt[1] := CHR(27);
   delay :=0;
END;
                                (* Type of cursor positioning *)
                                (* VT52, VT100 *)
curpos_alg := VT100;
                                (* True if line pos sent first *)
curpos_line_first := TRUE;
                                (* Offset for line, column *)
curpos_line_offset := 0;
curpos_col_offset :=0;
                                (* Beginning string of cursor positionning *)
WITH curpos_lead DO BEGIN
   len := 2;
   txt := '$[     ';
   txt[1] := CHR(27);
   delay :=0;
END;
                                (* Middle string of cursor positionning *)
WITH curpos_middle DO BEGIN
```

```
   len := 1;
   txt := '      ';
   delay :=0;
END;
                              (* Final string of cursor positionning *)
WITH curpos_end DO BEGIN
   len := 1;
   txt := 'H     ';
   delay :=0;
END;
                              (* Delay aftr cursor positionning, millisecs *)
curpos_delay := 0;
                              (* Start reverse *)
WITH start_reverse DO BEGIN
   len := 7;
   txt := '$[7;31m';
   txt[1] := CHR(27);
   delay :=0;
END;
                              (* Start blink *)
WITH start_blink DO BEGIN
   len := 0;
   txt := '      ';
   delay :=0;
END;
                              (* Start under *)
WITH start_under DO BEGIN
   len := 0;
   txt := '      ';
   delay :=0;
END;
                              (* Start bold *)
WITH start_bold DO BEGIN
   len := 0;
   txt := '      ';
   delay :=0;
END;
                              (* End high visibility *)
WITH end_highlight DO BEGIN
   len := 3;
   txt := '$[m   ';
   txt[1] := CHR(27);
   delay :=0;
END;
                              (* Cursor up *)
WITH cur_up DO BEGIN
   len := 3;
   txt :='$[A    ';
   txt[1] := CHR(27);
   delay := 0;
END;
                              (* Cursor down *)
WITH cur_do DO BEGIN
   len := 3;
   txt :='$[B    ';
   txt[1] := CHR(27);
   delay := 0;
END;
                              (* Cursor forward *)
WITH cur_fo DO BEGIN
```

```
      len := 3;
      txt :='$[C    ';
      txt[1] := CHR(27);
      delay := 0;
   END;
                                    (* Cursor backward *)
   WITH cur_ba DO BEGIN
      len := 3;
      txt :='$[D    ';
      txt[1] := CHR(27);
      delay := 0;
   END;
END; (* with sc *)
                                    (* Now, write this term def out *)
WRITE (sf, sc);
WRITELN (TTY, 'Stored definition for ', sc.id1,
      ',', sc.id2, ',', sc.id3, '.');


(* END GIGI *)
```

```
(* START --GIGIDEMO-- *)

WITH sc DO BEGIN
                                (* screen id, 15 chars, UPPERCASE *)
   id1 := 'GIGIDEMO        ';
   id2 := '                ';
   id3 := '                ';
                                (* height, width *)
   height := 24;
   width := 80;
                                (* Screen initialization - CSEQ 30 *)
   init_screen[1] := 27;         (* esc *)
   init_screen[2] := 91;         (* [ *)
   init_screen[3] := 63;         (* ? *)
   init_screen[4] := 55;         (* 7 *)
   init_screen[5] := 108;        (* l *)
   init_screen[6] := 0;
                                (* Screen reset - CSEQ 30 *)
   reset_screen[1] := 0;
                                (* Put cursor home *)
   WITH home DO BEGIN
      len := 4;
      txt := '$[;H    ';
      txt[1] := CHR(27);
      delay :=0;
   END;
                                (* Clear screen *)
   WITH clear_screen DO BEGIN
      len := 4;
      txt := '$[2J    ';
      txt[1] := CHR(27);
      delay :=0;
   END;
                                (* Clear to end of line *)
   WITH clear_eol DO BEGIN
      len := 3;
      txt := '$[K     ';
      TXT[1] := CHR(27);
      delay :=0;
   END;

   WITH clear_eos DO BEGIN
      len := 3;
      txt := '$[J     ';
      txt[1] := CHR(27);
      delay :=0;
   END;
                                (* Type of cursor positioning *)
                                (* VT52, VT100 *)
   curpos_alg := VT100;
                                (* True if line pos sent first *)
   curpos_line_first := TRUE;
                                (* Offset for line, column *)
   curpos_line_offset := 0;
   curpos_col_offset :=0;
                                (* Beginning string of cursor positionning *)
   WITH curpos_lead DO BEGIN
      len := 2;
      txt := '$[      ';
```

```
   txt[1] := CHR(27);
   delay :=0;
END;
                                (* Middle string of cursor positionning *)
WITH curpos_middle DO BEGIN
   len := 1;
   txt := ';      ';
   delay :=0;
END;
                                (* Final string of cursor positionning *)
WITH curpos_end DO BEGIN
   len := 1;
   txt := 'H      ';
   delay :=0;
END;
                                (* Delay aftr cursor positionning, millisecs *)
curpos_delay := 0;
                                (* Start reverse *)
WITH start_reverse DO BEGIN
   len := 7;
   txt := '$[7;31m';
   txt[1] := CHR(27);
   delay :=0;
END;
                                (* Start blink *)
WITH start_blink DO BEGIN
   len := 0;
   txt := '       ';
   delay :=0;
END;
                                (* Start under *)
WITH start_under DO BEGIN
   len := 0;
   txt := '       ';
   delay :=0;
END;
                                (* Start bold *)
WITH start_bold DO BEGIN
   len := 0;
   txt := '       ';
   delay :=0;
END;
                                (* End high visibility *)
WITH end_highlight DO BEGIN
   len := 3;
   txt := '$[m    ';
   txt[1] := CHR(27);
   delay :=0;
END;
                                (* Cursor up *)
WITH cur_up DO BEGIN
   len := 3;
   txt :='$[A    ';
   txt[1] := CHR(27);
   delay := 0;
END;
                                (* Cursor down *)
WITH cur_do DO BEGIN
   len := 3;
   txt :='$[B    ';
```

```
      txt[1] := CHR(27);
      delay := 0;
   END;
                                    (* Cursor forward *)
   WITH cur_fo DO BEGIN
      len := 3;
      txt :='$[C    ';
      txt[1] := CHR(27);
      delay := 0;
   END;
                                    (* Cursor backward *)
   WITH cur_ba DO BEGIN
      len := 3;
      txt :='$[D    ';
      txt[1] := CHR(27);
      delay := 0;
   END;
END; (* with sc *)
                                    (* Now, write this term def out *)
WRITE (sf, sc);
WRITELN (TTY, 'Stored definition for ', sc.id1,
      ',', sc.id2, ',', sc.id3, '.');


(* END GIGIDEMO *)
```

```
(* START --SIRIUS-- *)

WITH sc DO BEGIN
                                (* screen id, 15 chars, UPPERCASE *)
   id1 := 'SIRIUS          ';
   id2 := '                ';
   id3 := '                ';
                                (* height, width *)
   height := 24;
   width := 80;

                                (* Screen initialization - CSEQ 30 *)
   init_screen[1] := 0;
                                (* Screen reset - CSEQ 30 *)
   reset_screen[1] := 0;
                                (* Put cursor home *)
   WITH home DO BEGIN
      len := 3;
      txt := 's[H    ';
      txt[1] := CHR(27);
      delay :=0;
   END;
                                (* Clear screen *)
   WITH clear_screen DO BEGIN
      len := 6;
      txt := 's[Hs[J ';
      txt[1] := CHR(27);
      txt[4]:= CHR(27);
      delay :=0;
   END;
                                (* Clear to end of line *)
WITH clear_eol DO BEGIN
   len := 3;
   txt := 's[K    ';
   TXT[1] := CHR(27);
   delay :=0;
END;

WITH clear_eos DO BEGIN
   len := 3;
   txt := 's[J    ';
   txt[1] := CHR(27);
   delay :=0;
END;
                                (* Type of cursor positioning *)
                                (* VT52, VT100 *)
curpos_alg := VT100;
                                (* True if line pos sent first *)
curpos_line_first := TRUE;
                                (* Offset for line, column *)
curpos_line_offset := 0;
curpos_col_offset :=0;
                                (* Beginning string of cursor positionning *)
WITH curpos_lead DO BEGIN
   len := 2;
   txt := 's[     ';
   txt[1] := CHR(27);
   delay :=0;
END;
                                (* Middle string of cursor positionning *)
```

```
WITH curpos_middle DO BEGIN
   len := 1;
   txt := '      ';
   delay :=0;
END;
                                (* Final string of cursor positionning *)
WITH curpos_end DO BEGIN
   len := 1;
   txt := 'H     ';
   delay :=0;
END;
                                (* Delay aftr cursor positionning, millisecs *)
curpos_delay := 0;
                                (* Start reverse *)
WITH start_reverse DO BEGIN
   len := 4;
   txt := '$[7m   ';
   txt[1] := CHR(27);
   delay :=0;
END;
                                (* Start blink *)
WITH start_blink DO BEGIN
   len := 0;
   txt := '      ';
   delay :=0;
END;
                                (* Start under *)
WITH start_under DO BEGIN
   len := 0;
   txt := '      ';
   delay :=0;
END;
                                (* Start bold *)
WITH start_bold DO BEGIN
   len := 0;
   txt := '      ';
   delay :=0;
END;
                                (* End high visibility *)
WITH end_highlight DO BEGIN
   len := 3;
   txt := '$[m    ';
   txt[1] := CHR(27);
   delay :=0;
END;
                                (* Cursor up *)
WITH cur_up DO BEGIN
   len := 3;
   txt :='$[A    ';
   txt[1] := CHR(27);
   delay := 0;
END;
                                (* Cursor down *)
WITH cur_do DO BEGIN
   len := 3;
   txt :='$[B    ';
   txt[1] := CHR(27);
   delay := 0;
END;
                                (* Cursor forward *)
```

```
    WITH cur_fo DO BEGIN
       len := 3;
       txt :='$[C    ';
       txt[1] := CHR(27);
       delay := 0;
    END;
                                    (* Cursor backward *)
    WITH cur_ba DO BEGIN
       len := 3;
       txt :='$[D    ';
       txt[1] := CHR(27);
       delay := 0;
    END;
END; (* with sc *)
                                    (* Now, write this term def out *)
WRITE (sf, sc);
WRITELN (TTY, 'Stored definition for ', sc.id1,
       ',', sc.id2, ',', sc.id3, '.');


(* END SIRIUS *)
```

```
(* START VISUAL200 *)

WITH sc DO BEGIN
                                (* screen id, 15 chars, UPPERCASE *)
   id1 := 'VISUAL200       ';
   id2 := 'V200            ';
   id3 := 'VIS200          ';
                                (* height, width *)
   height := 24;
   width := 80;
                                (* Screen initialization - CSEQ 30 *)
   init_screen[1] := 27;
   init_screen[2] := 52;
   init_screen[3] := 0;
                                (* Screen reset - CSEQ 30 *)
   reset_screen[1] := 27;
   reset_screen[2] := 51;
   reset_screen[3] := 0;
                                (* Put cursor home *)
   WITH home DO BEGIN
      len := 4;
      txt := '$Y      ';
      txt[1] := CHR(27);
      delay :=0;
   END;
                                (* Clear screen *)
   WITH clear_screen DO BEGIN
      len := 6;
      txt := '$Y  $y ';
      txt[1] := CHR(27);
      txt[5] := CHR(27);
      delay :=0;
   END;
                                (* Clear to end of line *)
   WITH clear_eol DO BEGIN
      len := 2;
      txt := '$x      ';
      txt[1] := CHR(27);
      delay :=0;
   END;

   WITH clear_eos DO BEGIN
      len := 2;
      txt := '$y      ';
      txt[1] := CHR(27);
      delay :=0;
   END;
                                (* Type of cursor positioning *)
                                (* VT52, VT100 *)
   curpos_alg := VT52;
                                (* True if line pos sent first *)
   curpos_line_first := TRUE;
                                (* Offset for line, column *)
   curpos_line_offset := 32;
   curpos_col_offset := 32;
                                (* Beginning string of cursor positionning *)
   WITH curpos_lead DO BEGIN
      len := 2;
      txt := '$Y      ';
```

```
   txt[1] := CHR(27);
   delay :=0;
END;
                                (* Middle string of cursor positionning *)
WITH curpos_middle DO BEGIN
   len := 0;
   txt := '        ';
   delay :=0;
END;
                                (* Final string of cursor positionning *)
WITH curpos_end DO BEGIN
   len := 0;
   txt := '        ';
   delay :=0;
END;
                                (* Delay aftr cursor positionning, millisecs *)
curpos_delay := 0;
                                (* Start reverse *)
WITH start_reverse DO BEGIN
   len := 2;
   txt := '$3      ';
   txt[1] := CHR(27);
   delay :=0;
END;
                                (* Start blink *)
WITH start_blink DO BEGIN
   len := 0;
   txt := '        ';
   delay :=0;
END;
                                (* Start under *)
WITH start_under DO BEGIN
   len := 0;
   txt := '        ';
   delay :=0;
END;
                                (* Start bold *)
WITH start_bold DO BEGIN
   len := 0;
   txt := '        ';
   delay :=0;
END;
                                (* End high visibility *)
WITH end_highlight DO BEGIN
   len := 2;
   txt := '$4      ';
   txt[1] := CHR(27);
   delay :=0;
END;
                                (* Cursor up *)
WITH cur_up DO BEGIN
   len := 2;
   txt :='$A       ';
   txt[1] := CHR(27);
   delay := 0;
END;
                                (* Cursor down *)
WITH cur_do DO BEGIN
   len := 2;
   txt :='$B       ';
```

```
      txt[1] := CHR(27);
      delay := 0;
   END;
                                (* Cursor forward *)
   WITH cur_fo DO BEGIN
      len := 2;
      txt :='$C      ';
      txt[1] := CHR(27);
      delay := 0;
   END;
                                (* Cursor backward *)
   WITH cur_ba DO BEGIN
      len := 2;
      txt :='$D      ';
      txt[1] := CHR(27);
      delay := 0;
   END;
END; (* with sc *)
                                (* Now, write this term def out *)

WRITE (sf, sc);
WRITELN (TTY, 'Stored definition for ', sc.id1,
      ',', sc.id2, ',', sc.id3, '.');


(* END VISUAL200 *)
```

```
(* START CIT80 *)

WITH sc DO BEGIN
                                    (* screen id, 15 chars, UPPERCASE *)
   id1 := 'CIT80           ';
   id2 := '                ';
   id3 := '                ';
                                    (* height, width *)
   height := 24;
   width := 80;
                                    (* Screen initialization - CSEQ 30 *)
   init_screen[1] := 27;
   init_screen[2] := 60;
   init_screen[3] := 0;
                                    (* Screen reset - CSEQ 30 *)
   reset_screen[1] := 0;
                                    (* Put cursor home *)
   WITH home DO BEGIN
      len := 4;
      txt := '$Y      ';
      txt[1] := CHR(27);
      delay :=0;
   END;
                                    (* Clear screen *)
   WITH clear_screen DO BEGIN
      len := 6;
      txt := '$Y  $J  ';
      txt[1] := CHR(27);
      txt[5] := CHR(27);
      delay :=0;
   END;
                                    (* Clear to end of line *)
   WITH clear_eol DO BEGIN
      len := 2;
      txt := '$K      ';
      txt[1] := CHR(27);
      delay :=0;
   END;

   WITH clear_eos DO BEGIN
      len := 2;
      txt := '$J      ';
      txt[1] := CHR(27);
      delay :=0;
   END;
                                    (* Type of cursor positioning *)
                                    (* VT52, VT100 *)
   curpos_alg := VT52;
                                    (* True if line pos sent first *)
   curpos_line_first := TRUE;
                                    (* Offset for line, column *)
   curpos_line_offset := 32;
   curpos_col_offset := 32;
                                    (* Beginning string of cursor positionning *)
   WITH curpos_lead DO BEGIN
      len := 2;
      txt := '$Y      ';
      txt[1] := CHR(27);
      delay :=0;
```

```
END;
                                (* Middle string of cursor positionning *)
WITH curpos_middle DO BEGIN
   len := 0;
   txt := '        ';
   delay :=0;
END;
                                (* Final string of cursor positionning *)
WITH curpos_end DO BEGIN
   len := 0;
   txt := '        ';
   delay :=0;
END;
                                (* Delay aftr cursor positionning, millisecs *)
curpos_delay := 0;
                          (* Start reverse *)
WITH start_reverse DO BEGIN
   len := 0;
   txt := '        ';
   delay :=0;
END;
                                (* Start blink *)
WITH start_blink DO BEGIN
   len := 0;
   txt := '        ';
   delay :=0;
END;
                                (* Start under *)
WITH start_under DO BEGIN
   len := 0;
   txt := '        ';
   delay :=0;
END;
                                (* Start bold *)
WITH start_bold DO BEGIN
   len := 0;
   txt := '        ';
   delay :=0;
END;
                                (* End high visibility *)
WITH end_highlight DO BEGIN
   len := 0;
   txt := '        ';
   delay :=0;
END;
                                (* Cursor up *)
WITH cur_up DO BEGIN
   len := 2;
   txt :='$A      ';
   txt[1] := CHR(27);
   delay := 0;
END;
                                (* Cursor down *)
WITH cur_do DO BEGIN
   len := 2;
   txt :='$B      ';
   txt[1] := CHR(27);
   delay := 0;
END;
                                (* Cursor forward *)
```

```
      WITH cur_fo DO BEGIN
         len := 2;
         txt :='$C      ';
         txt[1] := CHR(27);
         delay := 0;
      END;
                                    (* Cursor backward *)
      WITH cur_ba DO BEGIN
         len := 2;
         txt :='$D      ';
         txt[1] := CHR(27);
         delay := 0;
      END;
   END; (* with sc *)
                                    (* Now, write this term def out *)

   WRITE (sf, sc);
   WRITELN (TTY, 'Stored definition for ', sc.id1,
          ',', sc.id2, ',', sc.id3, '.');


   (* END cit80 *)
```

```
(* START BEE1 *)

WITH sc DO BEGIN
                                (* screen id, 15 chars, UPPERCASE *)
   id1 := 'BEE1            ';
   id2 := 'MICROBEE1       ';
   id3 := '                ';
                                (* height, width *)
   height := 24;
   width := 80;
                                (* Screen initialization - CSEQ 30 *)
   init_screen[1] := 0;
                                (* Screen reset - CSEQ 30 *)
   reset_screen[1] := 0;
                                (* Put cursor home *)
   WITH home DO BEGIN
      len := 4;
      txt := '$Y      ';
      txt[1] := CHR(27);
      delay :=0;
   END;
                                (* Clear screen *)
   WITH clear_screen DO BEGIN
      len := 6;
      txt := '$Y  $J  ';
      txt[1] := CHR(27);
      txt[5] := CHR(27);
      delay :=0;
   END;
                                (* Clear to end of line *)
WITH clear_eol DO BEGIN
   len := 2;
   txt := '$K      ';
   txt[1] := CHR(27);
   delay :=0;
END;

WITH clear_eos DO BEGIN
   len := 2;
   txt := '$J      ';
   txt[1] := CHR(27);
   delay :=0;
END;
                                (* Type of cursor positioning *)
                                (* VT52, VT100 *)
curpos_alg := VT52;
                                (* True if line pos sent first *)
curpos_line_first := TRUE;
                                (* Offset for line, column *)
curpos_line_offset := 32;
curpos_col_offset := 32;
                                (* Beginning string of cursor positionning *)
WITH curpos_lead DO BEGIN
   len := 2;
   txt := '$Y      ';
   txt[1] := CHR(27);
   delay :=0;
END;
                                (* Middle string of cursor positionning *)
```

```
WITH curpos_middle DO BEGIN
   len := 0;
   txt := '       ';
   delay :=0;
END;
                                (* Final string of cursor positionning *)
WITH curpos_end DO BEGIN
   len := 0;
   txt := '       ';
   delay :=0;
END;
                                (* Delay aftr cursor positionning, millisecs *)
curpos_delay := 0;
                                (* Start reverse *)
WITH start_reverse DO BEGIN
   len := 3;
   txt := 'sdP    ';
   txt[1] := CHR(27);
   delay :=0;
END;
                                (* Start blink *)
WITH start_blink DO BEGIN
   len := 0;
   txt := '       ';
   delay :=0;
END;
                                (* Start under *)
WITH start_under DO BEGIN
   len := 0;
   txt := '       ';
   delay :=0;
END;
                                (* Start bold *)
WITH start_bold DO BEGIN
   len := 0;
   txt := '       ';
   delay :=0;
END;
                                (* End high visibility *)
WITH end_highlight DO BEGIN
   len := 3;
   txt := 'sd@    ';
   txt[1] := CHR(27);
   delay :=0;
END;
                                (* Cursor up *)
WITH cur_up DO BEGIN
   len := 2;
   txt :='$A      ';
   txt[1] := CHR(27);
   delay := 0;
END;
                                (* Cursor down *)
WITH cur_do DO BEGIN
   len := 2;
   txt :='$B      ';
   txt[1] := CHR(27);
   delay := 0;
END;
                                (* Cursor forward *)
```

```
    WITH cur_fo DO BEGIN
       len := 2;
       txt :='$C      ';
       txt[1] := CHR(27);
       delay := 0;
    END;
                                    (* Cursor backward *)
    WITH cur_ba DO BEGIN
       len := 2;
       txt :='$D      ';
       txt[1] := CHR(27);
       delay := 0;
    END;
END; (* with sc *)
                                    (* Now, write this term def out *)

WRITE (sf, sc);
WRITELN (TTY, 'Stored definition for ', sc.id1,
       ',', sc.id2, ',', sc.id3, '.');


(* END BEE1 *)
```

```
(* START MICROBEE2 *)

WITH sc DO BEGIN
                                (* screen id, 15 chars, UPPERCASE *)
   id1 := 'MICROBEE2       ';
   id2 := 'BEE2            ';
   id3 := '                ';
                                (* height, width *)
   height := 24;
   width := 80;
                                (* Screen initialization - CSE0 30 *)
   init_screen[1] := 0;
                                (* Screen reset - CSE0 30 *)
   reset_screen[1] := 0;
                                (* Put cursor home *)
   WITH home DO BEGIN
      len := 4;
      txt := '$Y      ';
      txt[1] := CHR(27);
      delay :=0;
   END;
                                (* Clear screen *)
   WITH clear_screen DO BEGIN
      len := 6;
      txt := '$Y  $J  ';
      txt[1] := CHR(27);
      txt[5] := CHR(27);
      delay :=0;
   END;
                                (* Clear to end of line *)
WITH clear_eol DO BEGIN
   len := 2;
   txt := '$K      ';
   txt[1] := CHR(27);
   delay :=0;
END;

WITH clear_eos DO BEGIN
   len := 2;
   txt := '$J      ';
   txt[1] := CHR(27);
   delay :=0;
END;
                                (* Type of cursor positioning *)
                                (* VT52, VT100 *)
curpos_alg := VT52;
                                (* True if line pos sent first *)
curpos_line_first := TRUE;
                                (* Offset for line, column *)
curpos_line_offset := 32;
curpos_col_offset := 32;
                                (* Beginning string of cursor positionning *)
WITH curpos_lead DO BEGIN
   len := 2;
   txt := '$Y      ';
   txt[1] := CHR(27);
   delay :=0;
END;
                                (* Middle string of cursor positionning *)
```

```
WITH curpos_middle DO BEGIN
   len := 0;
   txt := '        ';
   delay :=0;
END;
                                (* Final string of cursor positionning *)
WITH curpos_end DO BEGIN
   len := 0;
   txt := '        ';
   delay :=0;
END;
                                (* Delay aftr cursor positionning, millisecs *)
curpos_delay := 0;
                                (* Start reverse *)
WITH start_reverse DO BEGIN
   len := 3;
   txt := '$dP     ';
   txt[1] := CHR(27);
   delay :=0;
END;
                                (* Start blink *)
WITH start_blink DO BEGIN
   len := 0;
   txt := '        ';
   delay :=0;
END;
                                (* Start under *)
WITH start_under DO BEGIN
   len := 0;
   txt := '        ';
   delay :=0;
END;
                                (* Start bold *)
WITH start_bold DO BEGIN
   len := 0;
   txt := '        ';
   delay :=0;
END;
                                (* End high visibility *)
WITH end_highlight DO BEGIN
   len := 0;
   txt := '$d@     ';
   txt[1] := CHR(27);
   delay :=0;
END;
                                (* Cursor up *)
WITH cur_up DO BEGIN
   len := 2;
   txt :='$A      ';
   txt[1] := CHR(27);
   delay := 0;
END;
                                (* Cursor down *)
WITH cur_do DO BEGIN
   len := 2;
   txt :='$B      ';
   txt[1] := CHR(27);
   delay := 0;
END;
                                (* Cursor forward *)
```

```
   WITH cur_fo DO BEGIN
      len := 2;
      txt :='$C      ';
      txt[1] := CHR(27);
      delay := 0;
   END;
                                    (* Cursor backward *)
   WITH cur_ba DO BEGIN
      len := 2;
      txt :='$D      ';
      txt[1] := CHR(27);
      delay := 0;
   END;
END; (* with sc *)
                                    (* Now, write this term def out *)

WRITE (sf, sc);
WRITELN (TTY, 'Stored definition for ', sc.id1,
       ',', sc.id2, ',', sc.id3, '.');


(* END MICROBEE2 *)

                                    (* Now, CLOSE FILE *)
close (sf);

END.
```

UTILIT

```
{$M-}

TYPE char_set = SET OF CHAR;

{=============================================================================}

FUNCTION confirm(c:CHAR):BOOLEAN;
{ IN : c : caractere
  valeur de la fonction : booleen
  semantique : si "c" est 'v', 'Y', 'o' ou 'O', renvoie TRUE
               sinon, renvoie FALSE
}
BEGIN
   IF (c = 'v') OR (c = 'Y') OR (c = 'o') OR (c = 'O') THEN confirm :=TRUE
                                                       ELSE confirm :=FALSE;
END;

{-----------------------------------------------------------------------------}

PROCEDURE bell;
{ semantique : actionne le "beeper" du terminal }
BEGIN
   WRITE(TTY,CHR(7));
END;

{-----------------------------------------------------------------------------}

PROCEDURE getc(VAR c: CHAR);
{ IN : introduction au terminal
  OUT : c : caractere
  semantique : prend un caractere 'au vol' en garantissant un huitieme
               bit a 0
}
VAR toto:integer;

BEGIN
   jsys(73b;;c);
   toto := ORD(c);
   IF toto > 128 THEN toto := toto - 128;
   c := CHR(toto);
END;

{-----------------------------------------------------------------------------}

FUNCTION n_to_s(i : INTEGER):CHAR;

{ IN : i : entier entre 0 et 9
  valeur de la fonction : caractere entre '0' et '9'
  semantique : convertit un chiffre entre 0 et 9 en un caractere entre '0'
               et '9'
}

VAR c : CHAR;

BEGIN
   c := ' ';
   IF (i >= 0) AND (i <= 9) THEN c := CHR(48 + i);
   n_to_s := c;
END;
```

```
{-------------------------------------------------------------------------------}

FUNCTION s_to_n(VAR str : PACKED ARRAY [m..n:INTEGER] OF CHAR): INTEGER;

{ IN : chaine de caracteres numeriques
  valeur de la fonction : entier
  semantique : convertit un string de CARACTERES numeriques en un ENTIER
}

VAR i,total: INTEGER;
    fini : BOOLEAN;

BEGIN
   fini := FALSE;
   total := 0 ;
   i := 1;
   WHILE (str[i] >= '0') AND (str[i] <= '9') AND NOT fini DO
    BEGIN
      total := (total * 10) + (ORD(str[i]) - ORD('0'));
      IF (i + 1) > n THEN fini := TRUE ELSE i := i + 1;
    END;
   s_to_n := total;
END;

{-------------------------------------------------------------------------------}

FUNCTION erstat(VAR f:FILE):INTEGER ;EXTERN;

{ IN : f : nom interne de fichier
  valeur de la fonction : entier
  semantique : vaut 0 si l'ouverture du fichier de nom "f" s'est passee
               sans erreur;
               indique le numero de l'erreur (tel que defini dans
               l'appel systeme TOPS-20 (JSYS) GETER ) s'il y a eu erreur
}

{-------------------------------------------------------------------------------}

FUNCTION alpha(VAR str : PACKED ARRAY[m..n:INTEGER] OF CHAR): BOOLEAN;

{ IN : str : chaine de caracteres
  valeur de la fonction : booleen
  semantique : retourne FALSE si "str" contient des caracteres numeriques,
               TRUE sinon
}

VAR alphabetic : SET OF CHAR;
    fini : BOOLEAN;
    i : INTEGER;

BEGIN
   fini := FALSE;
   alphabetic := [' '..'/',':'..'~'];
   alpha := TRUE;
   i := m;
   WHILE (i < n) AND NOT fini DO
      BEGIN
         IF NOT (str[i] IN alphabetic) THEN
            BEGIN
               fini := TRUE;
```

```
                alpha := FALSE;
              END;
            i := i + 1;
        END;
END;

{-------------------------------------------------------------------------------}

FUNCTION num(VAR str : PACKED ARRAY [m..n:INTEGER] OF CHAR):BOOLEAN;

{ IN : str : chaine de caracteres
  valeur de la fonction : booleen
  semantique : retourne TRUE si "str" est compose uniquement de chiffres,
                        FALSE sinon
}

VAR numeric : SET OF CHAR;
    fini : BOOLEAN;
    i : INTEGER;

BEGIN
   fini := FALSE;
   numeric := ['0'..'9',' '];
   num := TRUE;
   i := m;
   WHILE (i < n) AND NOT fini DO
      BEGIN
         IF NOT (str[i] IN numeric) THEN
            BEGIN
               fini := TRUE;
               num := FALSE;
            END;
         i := i + 1;
      END;
END;

{-------------------------------------------------------------------------------}

FUNCTION match(VAR pattern  : PACKED ARRAY [m..n:INTEGER] OF CHAR;
                   VAR str : PACKED ARRAY [p..q:INTEGER] OF CHAR):BOOLEAN;

{ IN : pattern, str : chaines de caracteres
  valeur de la fonction : booleen
  semantique : retourne TRUE si les n caracteres du  string "pattern"
               sont identiques aux n premiers caracteres du string "str",
               FALSE sinon
}

VAR diff,fini : BOOLEAN;
    i,j : INTEGER;

BEGIN
   diff := FALSE;
   fini := FALSE;
   i := m ; j := p;
   WHILE NOT diff AND NOT fini DO
      BEGIN
         IF str[i] <> pattern[j] THEN diff := TRUE;
         i := i + 1;
         j := j + 1;
```

```
        IF (i > n) OR (i > o) THEN fini := TRUE
                             ELSE IF (str[i] =' ' ) OR (pattern[i] = ' ')
                                                   THEN fini := TRUE;
      END;
   match := NOT diff
END;

{-------------------------------------------------------------------------}

PROCEDURE wstr(VAR fich : FILE ;VAR str : PACKED ARRAY [m..n:INTEGER] OF CHAR;
               VAR nb_char : INTEGER );
{ IN : fich : nom interne de fichier
       str : chaine de caracteres
  OUT : nb_car : entier
  semantique : ecrit le string "str" sur le fichier "fich" sans les blancs
               de terminaison.
               "nb_char" contient le nombre de caracteres effectivement
               ecrits
}

BEGIN
   nb_char := n;
   WHILE (str[nb_char] = ' ') AND (nb_char > m) DO nb_char := nb_char - 1;
   WRITE(fich,str:nb_char);
END;

{-------------------------------------------------------------------------}

PROCEDURE discard(garbage : char_set ;VAR fich : TEXT);
{ IN : garbage : ensemble de caracteres
       fich : nom interne d'un fichier de caracteres
  semantique : lit des caracteres dans le fichier "fich" tant qu'il s'agit
               de caracteres de l'ensemble "garbage"
}

BEGIN
   WHILE fich^ IN garbage DO GET(fich);
END;

{-------------------------------------------------------------------------}

PROCEDURE lower(VAR str,res : PACKED ARRAY [m..n:INTEGER] OF CHAR);
{ IN : str : chaines de caracteres
  OUT : res : chaines de caracteres
  semantique : copie "str" dans "res" en convertissant les lettres majuscules
               en lettres minuscules
}

VAR i : INTEGER;

BEGIN
   FOR i:= m TO n DO
      IF (ORD(str[i]) > 64) AND (ORD(str[i]) < 91 )
         THEN res[i] := CHR(ORD(str[i]) + 32)
         ELSE res[i] := str[i];
END;

{-------------------------------------------------------------------------}

PROCEDURE concat(VAR str1 : PACKED ARRAY [b_str1..e_str1: INTEGER] OF CHAR;
```

```
                VAR str2 : PACKED ARRAY [b_str2..e_str2: INTEGER] OF CHAR;
                VAR res : PACKED ARRAY [b_res..e_res: INTEGER] OF CHAR);
{ IN : str1, str2 : chaines de caracteres
  OUT : res : chaine de caracteres
  semantique : concatene les strings "str1" et "str2" en un string "res".
               Si les longueurs ne correspondent pas, il y a troncature ou
               completion par des blancs
}

LABEL 99;

VAR i,j : INTEGER;
    fini : BOOLEAN;

BEGIN
   FOR i := b_res TO e_res DO res[i] :=' ';
   i := b_res;
   j := b_str1;
   fini := (str1[j]=' ');
   WHILE NOT fini DO
      BEGIN
         res[i] := str1[j];
         i := i + 1;
         j := j + 1;
         IF (i > e_res) THEN GOTO 99;
         IF (j > e_str1) THEN fini := TRUE
                         ELSE IF str1[j] = ' ' THEN fini := TRUE;
      END;
   j := b_str2;
   fini := (str2[j]=' ');
   WHILE NOT fini DO
      BEGIN
         res[i] := str2[j];
         i := i + 1;
         j := j + 1;
         IF (i > e_res) OR (j > e_str2) THEN fini := TRUE
                                        ELSE IF str2[j] = ' ' THEN fini := TRUE;
      END;
99:
END;

{-------------------------------------------------------------------------}

FUNCTION simili(VAR str1,str2 : PACKED ARRAY [m..n:INTEGER] OF CHAR):INTEGER;
{ IN : str1, str2 : chaines de caracteres
  semantique : fournit un entier indiquant le degre de similarite entre
               "str1" et "str2".
               Cet entier vaut : 0  si les deux strings sont egaux
                                 -1 si ils sont differents de plus de deux
                                    lettres consecutives
                                 n>0, avec n le nombre de 'differences' entre
                                    les deux strings. (par 'difference ' on
                                    entend deux lettres differentes mais
                                    suivies de deux lettres egales)
}

VAR i,j,diff : INTEGER;

BEGIN
   i:=0 ; j:=0 ;
```

```
   diff := 0;
   WHILE (i < n) AND (i < n) AND (diff >= 0) DO
      BEGIN
         i := i + 1;
         j := j + 1;
         IF str1[i] <> str2[j] THEN
            BEGIN
               diff := diff + 1;
               i := i + 1;
               IF str1[i] <> str2[j] THEN
                  BEGIN
                     i := i - 1;
                     j := j + 1;
                     IF str1[i] <> str2[j] THEN
                        BEGIN
                           i := i + 1;
                           IF str1[i] <> str2[j] THEN diff := -1;
                        END;
                  END;
            END;
      END;
   simili := diff;
END.
```

DACCES

```
{$M-}

include 'sys:string.dec'; { necesaire pour GETEDI }
include 'sys:cod_ret.dec';
include 'sys:utilit.dec';
include 'sys:tscr.dec';
include 'sys:getedi.dec';

{-------------------------------------------------------------------------}

PROCEDURE dacces(VAR fich : TEXT;sepa : INTEGER);
{ IN : fich : fichier de caracteres
       debut : numero de la ligne de l'ecran a partir de laquelle on desire
               que l'affichage se passe
       commandes de l'utilisateur au terminal
  OUT : affichage du contenu du fichier
  semantique : affiche le contenu du fichier "fich" sur l'ecran d'un
               terminal, a partir de la ligne "debut"
               L'utilisateur peut donner les commandes suivantes :
                     <space> pour avancer d'un ecran dans le fichier
                     <backspace> pour reculer d'un ecran
                     'q' pour terminer l'affichage
                     control-p pour imprimer le contenu du fichier
}

LABEL 100;

CONST over = 2; { zone de recouvrement entre deux ecrans }
      msg_suite = '<space> : avancer,<backspace> : reculer,''q'': quitter,<ctrl-p> : imprimer   ';

TYPE d_pos = ^position;
     position = RECORD
                prec : d_pos;
                cont : INTEGER;
                next : d_pos;
     END;


VAR c : CHAR;
    i,posit : INTEGER;
    lst_pos,cour,bid : d_pos;

    fenetre : window;

    code_ret : typ_code_ret;

    ligne_blanche : PACKED ARRAY [1..lim_col] OF CHAR;

    fout : TEXT;

{procedures internes }

PROCEDURE ins_pos(VAR tete : d_pos;contenu : INTEGER;VAR position_prec : d_pos);
{ IN : tete : designateur vers une liste de "positions"
       contenu : "position" a inserer dans la liste des "positions"
       position_prec : designateur vers un element de la liste des
                       "positions".
  OUT : liste des positions modifiee
  semantique : insere dans la liste designee par "tete", un element
               contenant "contenu", apres l'element designe par
```

```
                "position_prec"
                Si "position_prec" est vide, on insere en tete de liste.
}

VAR nouv_pos : d_pos;

BEGIN
   NEW(nouv_pos);
   nouv_pos^.cont := contenu;
   nouv_pos^.prec := position_prec;
   IF position_prec = NIL THEN
      BEGIN
         nouv_pos^.next := tete;
         tete := nouv_pos
      END
   ELSE
      BEGIN
         nouv_pos^.next := position_prec^.next;
         position_prec^.next := nouv_pos
      END;
END;

{-------------------------------------------------------------------------------}

PROCEDURE lect_par_fen(VAR curwin : window;VAR fich :TEXT;VAR posit : INTEGER);

{ IN : curwin : designateur vers une fenetre
       posit : entier
       fich : fichier de caracteres, ouvert en lecture, acces direct
  OUT : action sur l'affichage;
        curwin et posit modifiees
  parametre global : over : nombre de lignes de recouvrement entre deux
                            ecrans
                     mso_suite : chaine de caracteres de longueur 80;
  semantique : garnit les n - 2 premieres lignes de la fenetre "curwin"
               avec le contenu du fichier "fich", depuis l'endroit designe
               par "posit". La derniere ligne affiche le message donne
               dans "mso_suite".
               Apres l'appel, "posit" indique l'endroit du fichier "fich"
               qui a ete affiche a "over" lignes de la fin de l'ecran
}

VAR i,n_ligne : INTEGER;
    str : PACKED ARRAY [1..lim_col] OF CHAR;


BEGIN
   WITH curwin^ DO
      BEGIN
         setpos(fich,posit);
         n_ligne := maxx - begx - 2;
         i := 1;
         curx := begx;
         posit := -1;
         erase(curwin);
{        WHILE (i <= (n_ligne )) AND  NOT EOF(fich) do (conceptuellement)}
         WHILE i <= n_ligne DO
            BEGIN
               READ(fich,str);READLN(fich);
               curv := i;
```

```
                addstr(curwin,str,normal);
                IF i = (n_ligne - over) THEN posit := (curpos(fich)-1);
                i := i + 1 ;
                curx := curx + 1;
                IF EOF(fich) THEN
                   BEGIN
                      cury := 1;
                      WHILE i <= n_ligne DO
                         BEGIN
                            addstr(curwin,ligne_blanche,normal);
                            curx := curx + 1 ;
                            i := i + 1 ;
                         END;
                   END;
             END;
          curx := maxx;
          cury := 1;
          addstr(curwin,msg_suite,normal);
          refresh(curwin);
       END;
END;


{corps de la procedure dacces}
{-------------------------------------------------------------------------}


BEGIN
   FOR i := 1 TO lim_col DO ligne_blanche[i] := ' ';
   RESET(fich,'','/E/I');
   fenetre := newwin(sepa,1,xphysx,xphysy);
   lst_pos := NIL;
   bid := NIL;
   ins_pos(lst_pos,0,bid);
   cour := lst_pos;
   posit := 0;

   lect_par_fen(fenetre,fich,posit); { premiere lecture a partir de 0 }

   getc(c);
   WHILE c <> 'q' DO
      BEGIN
         CASE c OF
            ' ':BEGIN
                IF cour^.next = NIL
                   THEN IF EOF(fich) THEN aff_msg(fdf)
                                     ELSE BEGIN
                                             IF posit >= 0 THEN ins_pos(lst_pos,posit,cour);
                                             lect_par_fen(fenetre,fich,posit);
                                             cour := cour^.next;
                                          END
                   ELSE BEGIN
                           cour := cour^.next;
                           posit := cour^.cont;
                           lect_par_fen(fenetre,fich,posit);
                        END;
            END;
            '^H': BEGIN
               IF cour^.prec = NIL THEN aff_msg(ddf)
                                   ELSE cour := cour^.prec;
```

```
                posit := cour^.cont;
                lect_car_fen(fenetre,fich,posit);
             END;
          END { case };
          getc(c);
                              { on est oblige de tester le cas '^P' et le
                                cntrl_l avant le 'CASE' car il n'accepte
                                pas ce choix }
          IF c = CHR(16) THEN
             BEGIN
                REWRITE(fout,'lpt0:out.txt');
                RESET(fich,'','/E');
                WHILE NOT EOF(fich) DO
                   BEGIN
                      fout^:=fich^;
                      PUT(fout);
                      GET(fich);
                   END;
                aff_msg(impok);
             END
          ELSE IF c = CHR(12) THEN ctrl_l;

       END;
     clearall(fenetre);
     refresh(fenetre);
100:
END.
```

CONVDA

```
{$M-}
include 'sys:string.dec';
include 'sys:cod_ret.dec';
include 'sys:utilit.dec';


PROCEDURE conv_date(VAR date_in : PACKED ARRAY[m..n:INTEGER] OF CHAR;
                    VAR date_out : str10:VAR code_ret : typ_code_ret);
{ IN : date_in : date sous forme libre jour-mois-annee, en anglais ou en
                 francais .
  OUT : date_out : date sous forme interne AAAA/MM/JJ.
  semantique : recoit une date en "date_in" sous toute forme raisonnable
               telle que JOUR est avant MOIS est avant ANNEE (en anglais
               ou en francais), la verifie, et, si elle est correcte,
               la convertit en "date_out" sous forme interne (AAAA/MM/JJ)

      Valeurs possibles pour le code retour :
          Un parmi les codes de l'analyse syntaxique et semantique
          join : jour doit etre inferieur au jour courant
          moin : mois doit etre inferieur au mois courant
          anin : annee doit etre inferieure a l'annee courante

}


LABEL 100;

TYPE struct_date = RECORD
        jour : INTEGER;
        mois : INTEGER;
        annee : INTEGER;
     END;
     characters = SET OF CHAR;

VAR date_ok_synt : struct_date;
    date_syst : str9;
    date_syst_conv : struct_date;

{Procedures et fonctions internes a la procedure 'conv_date}
{-------------------------------------------------------------------------}

FUNCTION bissext(annee : INTEGER ):BOOLEAN;

{ IN : annee : entier
  valeur de la fonction : booleen
  semantique : retourne TRUE si l'annee (exprimee sous la forme
               "4 chiffres" !!! ) est bissextile , FALSE sinon}

BEGIN
   IF (((annee MOD 4) = 0) AND ((annee MOD 100) <> 0)) OR ((annee MOD 1000) = 0)
                    THEN bissext := TRUE
                    ELSE bissext := FALSE;
END;


PROCEDURE mois_str_to_mois_int(mois :str10 ; VAR  mois_int : INTEGER );

{ IN : mois : chaine de caracteres de longueur 10
  OUT : mois_int : entier
```

```
  semantique : transforme un nom de mois anglais ou francais en son
               equivalent 'numerique '; Cette procedure accepte toute
               abbreviation;

               Retourne 0 si pas trouve;

               Ne detecte pas les ambiguites !!! p.ex 'jui' pour juin et
                    juillet !!!
               (retourne la premiere correspondance trouvee)

}

VAR l_mois : str10;
BEGIN
   mois_int := 0;
   lower(mois,l_mois);
   IF match(l_mois,'janvier')
      OR match(l_mois,'january') THEN mois_int := 1
   ELSE IF match(l_mois,'fevrier')
         OR match(l_mois,'february') THEN mois_int := 2
   ELSE IF match(l_mois,'mars')
         OR match(l_mois,'march') THEN mois_int := 3
   ELSE IF match(l_mois,'avril')
         OR match(l_mois,'april') THEN mois_int := 4
   ELSE IF match(l_mois,'mai')
         OR match(l_mois,'may') THEN mois_int := 5
   ELSE IF match(l_mois,'juin')
         OR match(l_mois,'june') THEN mois_int := 6
   ELSE IF match(l_mois,'juillet')
         OR match(l_mois,'july') THEN mois_int := 7
   ELSE IF match(l_mois,'aout')
         OR match(l_mois,'august') THEN mois_int := 8
   ELSE IF match(l_mois,'septembre')
         OR match(l_mois,'september') THEN mois_int := 9
   ELSE IF match(l_mois,'octobre')
         OR match(l_mois,'october') THEN mois_int := 10
   ELSE IF match(l_mois,'novembre')
         OR match(l_mois,'november') THEN mois_int := 11
   ELSE IF match(l_mois,'decembre')
         OR match(l_mois,'december') THEN mois_int := 12;
END;

{-------------------------------------------------------------------------}

PROCEDURE an_svnt_et_conv(VAR date_in : PACKED ARRAY [m..n:INTEGER] OF CHAR;
                  VAR date_out : struct_date;VAR code_ret:typ_code_ret);

{ IN : date_in : chaine de carateres
  OUT : date_out : date sous forme interne
        cod_ret : code retour
  semantique : verifie si la chaine "date_in" correspond a une date sous
               forme 'JOUR puis MOIS puis ANNEE ' . Si c'est le cas, elle est
               convertie en une date sous forme interne "date_out". Sinon,
               le code retour indique l'erreur rencontree.

        Valeur du code retour :
          ok
          jnum : jour doit etre numerique
          lj2 : longueur du champ jour doit etre inferieure a 2
          lm2 : longueur du champ mois (numerique) doit etre inferieure a 2
```

```
          lm10 : longueur du champ mois (alpha) doit etre inferieure a 2
          mnc : mois non connu
          anum : annee doit etre numerique
}

LABEL 99;

VAR f_date : TEXT;
    jour : str2;
    mois : str10;
    annee : str4;
    annee_int : INTEGER;
    breakset : SET OF CHAR;
    long_jour,long_mois,long_annee : INTEGER;


BEGIN
   code_ret := ok;

   date_out.jour := 0;
   date_out.mois := 0;
   date_out.annee := 0;

   breakset := ['/',' ',',','-','_'];
   strset(f_date,date_in);
   discard(breakset,f_date);
   READ(f_date,jour:long_jour:breakset);
   discard(breakset,f_date);
   READ(f_date,mois:long_mois:breakset);
   discard(breakset,f_date);
   READ(f_date,annee:long_annee:breakset);


   IF NOT num(jour) THEN BEGIN code_ret := jnum ;GOTO 99 END;
   IF long_jour > 2 THEN BEGIN code_ret := lj2 ;GOTO 99 END
                    ELSE date_out.jour := s_to_n(jour);

   IF NOT alpha(mois)
     THEN BEGIN
            IF long_mois > 2 THEN BEGIN code_ret := lm2 ;GOTO 99 END;
            date_out.mois := s_to_n(mois);
          END
     ELSE BEGIN
            IF long_mois > 10 THEN BEGIN code_ret := lm10 ;GOTO 99 END;
            mois_str_to_mois_int(mois,date_out.mois);
            IF date_out.mois = 0 THEN BEGIN code_ret := mnc ;GOTO 99 END;
          END;

   IF NOT num(annee) THEN BEGIN code_ret := anum ; GOTO 99 END;
   annee_int := s_to_n(annee);
   IF annee_int < 100 THEN annee_int := annee_int + 1900;
   date_out.annee := annee_int;
      99:
END;

{-------------------------------------------------------------------------}

PROCEDURE an_sem(date_in : struct_date;VAR date_out : str10;VAR code_ret : typ_code_ret);
{ IN : date_in : date sous forme interne
  OUT : date_out : date sous forme AAAA/MM/JJ
```

```
  semantique : verifie si la date "date_in" est correcte semantiquement.
               si oui, retourne la date verifiee dans "date_out"
               sinon, le code retour "code_ret" indique l'erreur rencontree.

    Valeurs possibles du code retour :
       ok
       rmo : range mois incorrect (doit etre entre 1 et 12)
       ri28
       ri29
       ri30
       ri31 : range jour incorrect

}

LABEL 99;

VAR  f_date : TEXT;

BEGIN
   code_ret := ok;

   date_out := bl10;

   WITH date_in DO
      BEGIN
         IF (mois < 1) OR (mois > 12) THEN BEGIN code_ret := rmo ;GOTO 99 END;
         IF (jour < 1) OR (jour > 31) THEN BEGIN code_ret := ri31 ;GOTO 99 END;
         IF ((mois = 4) OR
             (mois = 6) OR
             (mois = 9) OR
             (mois = 11)) AND (jour > 30) THEN BEGIN code_ret := ri30;GOTO 99 END;
         IF (mois = 2) THEN
            BEGIN
              IF bissext(annee)
                 THEN IF (jour > 29) THEN BEGIN code_ret := ri29 ;GOTO 99 END ELSE
                 ELSE IF (jour > 28) THEN BEGIN code_ret := ri28 ; GOTO 99 END;
            END;

         strwrite(f_date,date_out);
         WRITE(f_date,annee:4,'/',mois:2,'/',jour:2);
      END;
    99:
END;


{corps de la procedure 'conv_date}
{-------------------------------------------------------------------------}


BEGIN
   an_synt_et_conv(date_in,date_ok_synt,code_ret);
   IF code_ret <> ok THEN GOTO 100;

   date_syst := date;
   an_synt_et_conv(date_syst,date_syst_conv,code_ret);
   IF date_ok_synt.annee > date_syst_conv.annee
      THEN BEGIN code_ret := anin ; GOTO 100 END ;
   IF date_ok_synt.annee = date_syst_conv.annee
      THEN IF date_ok_synt.mois > date_syst_conv.mois
           THEN BEGIN code_ret := moin ; GOTO 100 END;
```

```
   IF (date_ok_synt.annee = date_syst_conv.annee) AND
                        (date_ok_synt.mois = date_syst_conv.mois )
      THEN IF date_ok_synt.jour > date_syst_conv.jour
           THEN BEGIN code_ret := join; GOTO 100 END;
   an_sem(date_ok_synt,date_out,code_ret);
 100:
END;

{=============================================================================}


PROCEDURE pretty_date(date_in : str10 ;VAR  jolie_date : str20);
{ IN : date_in : date sous forme interne
  OUT : date_out : date sous une forme agreable a lire (jour, mois en toutes
                   lettres et annee en quatre chiffres).
  semantique : convertit la date (supposee valide) en une date agreable
               a lire.
}

VAR fstr : TEXT;
    jour,mois,annee : INTEGER;
    tab_mois : PACKED ARRAY [1..12] OF str10;
    long_mois : ARRAY[1..12] OF INTEGER;

BEGIN
   jolie_date := bl20;

   tab_mois[1] :='Janvier   ';long_mois[1] := 8;
   tab_mois[2] :='Fevrier   ';long_mois[2] := 8;
   tab_mois[3] :='Mars      ';long_mois[3] := 5;
   tab_mois[4] :='Avril     ';long_mois[4] := 6;
   tab_mois[5] :='Mai       ';long_mois[5] := 4;
   tab_mois[6] :='Juin      ';long_mois[6] := 5;
   tab_mois[7] :='Juillet   ';long_mois[7] := 8;
   tab_mois[8] :='Aout      ';long_mois[8] := 5;
   tab_mois[9] :='Septembre ';long_mois[9] := 10;
   tab_mois[10]:='Octobre   ';long_mois[10] := 8;
   tab_mois[11]:='Novembre  ';long_mois[11] := 9;
   tab_mois[12]:='Decembre  ';long_mois[12] := 9;
   strset(fstr,date_in);
   READ(fstr,annee);
   GET(fstr);
   READ(fstr,mois);
   GET(fstr);
   READ(fstr,jour);

   strwrite(fstr,jolie_date);
   WRITE(fstr,jour:2,' ',tab_mois[mois]:long_mois[mois],annee:4);
END.
```

SORT

```
{$M-}
include 'sys:string.dec';
include 'sys:elems.dec';
include 'sys:utilit.dec';

{TYPE str80 = PACKED ARRAY [1..80] OF CHAR;
     str20 = PACKED ARRAY [1..20] OF CHAR;
     str10 = PACKED ARRAY [1..10] OF CHAR;
     str9 = PACKED ARRAY [1..9] OF CHAR;
     str3 = PACKED ARRAY [1..3] OF CHAR;}


{PROCEDURE lower(VAR str,res : PACKED ARRAY [m..n:INTEGER] OF CHAR);}


{=============================================================================}
```

```
{ Les procedures X_sort sont compose'es de trois parties :
   - transfert des e'le'ments du fichier de X vers des ele'ments en me'moire
     centrale, les pointeurs vers ces e'le'ments e'tant regroupe' dans une
     table de pointeurs (proce'dure READ_IN_TAB)

   - tri de la table des pointeurs, la cle' e'tant la valeur de l'e'le'ment
     pointe'(proce'dure SORT)

   - transfert des e'le'ments pointe's par la table des pointeurs vers le
     fichier(proce'dure WRITE_FROM_TAB)
}

PROCEDURE fnom_sort(VAR fich : f_el_nom);

TYPE d_el_nom = ^el_nom;

VAR tab_a_trier : ARRAY[1..200] OF d_el_nom;
    nb_el : INTEGER;

{procedures internes a 'fnom_sort'}
{-------------------------------------------------------------------------}

PROCEDURE read_in_tab(VAR filin : f_el_nom;
              VAR tab : ARRAY[m..n:INTEGER] OF d_el_nom;VAR nb_el : INTEGER);

VAR nouv_el : d_el_nom;

BEGIN
   nb_el := 0;
   WHILE (NOT EOF(filin)) AND (nb_el < (n-m+1)) AND (nb_el < 200) DO
      BEGIN
         NEW(nouv_el);
         nouv_el^ := filin^;
         nb_el := nb_el + 1 ;
         tab[nb_el] := nouv_el;
         GET(filin);
      END;
END;


PROCEDURE write_from_tab(VAR tab : ARRAY[m..n:INTEGER] OF d_el_nom;
                         nb_el : INTEGER;VAR filout : f_el_nom);

VAR i : INTEGER;

BEGIN
   FOR i := 1 TO nb_el DO
      BEGIN
         WRITE(filout,tab[i]^);
         DISPOSE(tab[i]);
      END;
END;


PROCEDURE sort(VAR tab: ARRAY[m..n:INTEGER] OF d_el_nom; nb_el : INTEGER);

VAR i,j,ind_cour : INTEGER;
    l_el_min_cour_content,l_tabj_content : str20;
```

```
    d_el_min_cour : d_el_nom;

BEGIN
   FOR i := 1 TO nb_el - 1 DO
      BEGIN
         ind_cour := i;
         d_el_min_cour := tab[i];

         lower(d_el_min_cour^.content,l_el_min_cour_content);

         FOR j := i + 1 TO nb_el DO
            BEGIN
               lower (tab[j]^.content,l_tabj_content);
               IF l_tabj_content < l_el_min_cour_content THEN
                  BEGIN
                     ind_cour := j;
                     d_el_min_cour := tab[i];
                     lower(d_el_min_cour^.content,l_el_min_cour_content);
                  END;
            END;
         tab[ind_cour] := tab[i];
         tab[i] := d_el_min_cour;
      END;
END; {sort}

{corps de la procedure 'fnom_sort'}
{=============================================================================}

BEGIN
   RESET(fich);
   read_in_tab(fich,tab_a_trier,nb_el);
   sort(tab_a_trier,nb_el);
   REWRITE(fich);
   write_from_tab(tab_a_trier,nb_el,fich);
END;

{=============================================================================}
```

```
PROCEDURE fdate_sort(VAR fich : f_el_date);

TYPE d_el_date = ^el_date;

VAR tab_a_trier : ARRAY[1..200] OF d_el_date;
    nb_el : INTEGER;

{procedures internes a 'fdate_sort'}
{-------------------------------------------------------------------------}

PROCEDURE read_in_tab(VAR filin : f_el_date;
              VAR tab : ARRAY[m..n:INTEGER] OF d_el_date;VAR nb_el : INTEGER);

VAR nouv_el : d_el_date;

BEGIN
   nb_el := 0;
   WHILE (NOT EOF(filin)) AND (nb_el < (n-m+1)) AND (nb_el < 200) DO
      BEGIN
         NEW(nouv_el);
         nouv_el^ := filin^;
         nb_el := nb_el + 1 ;
         tab[nb_el] := nouv_el;
         GET(filin);
      END;
END;


PROCEDURE write_from_tab(VAR tab : ARRAY[m..n:INTEGER] OF d_el_date;
                         nb_el : INTEGER;VAR filout : f_el_date);

VAR i : INTEGER;

BEGIN
   FOR i := 1 TO nb_el DO
      BEGIN
         WRITE(filout,tab[i]^);
         DISPOSE(tab[i]);
      END;
END;


PROCEDURE sort(VAR tab: ARRAY[m..n:INTEGER] OF d_el_date; nb_el : INTEGER);

VAR i,j,ind_cour : INTEGER;
    l_el_min_cour_content,l_tabj_content : str10;
    d_el_min_cour : d_el_date;

BEGIN
   FOR i := 1 TO nb_el - 1 DO
      BEGIN
         ind_cour := i;
         d_el_min_cour := tab[i];

         lower(d_el_min_cour^.content,l_el_min_cour_content);

         FOR j := i + 1 TO nb_el DO
            BEGIN
```

```
                lower (tab[j]^.content,l_tabj_content);
                IF l_tabj_content < l_el_min_cour_content THEN
                   BEGIN
                      ind_cour := j;
                      d_el_min_cour := tab[j];
                      lower(d_el_min_cour^.content,l_el_min_cour_content);
                   END;
             END;
          tab[ind_cour] := tab[i];
          tab[i] := d_el_min_cour;
       END;
END; {sort}

{corps de la procedure 'fdate_sort'}
{-------------------------------------------------------------------------}

BEGIN
   RESET(fich);
   read_in_tab(fich,tab_a_trier,nb_el);
   sort(tab_a_trier,nb_el);
   REWRITE(fich);
   write_from_tab(tab_a_trier,nb_el,fich);
END;

{=========================================================================}
```

```
PROCEDURE fmc_sort(VAR fich : f_el_mc);

TYPE d_el_mc = ^el_mc;

VAR tab_a_trier : ARRAY[1..200] OF d_el_mc;
    nb_el : INTEGER;

{procedures internes a 'fmc_sort'}
{-------------------------------------------------------------------------}

PROCEDURE read_in_tab(VAR filin : f_el_mc;
                VAR tab : ARRAY[m..n:INTEGER] OF d_el_mc;VAR nb_el : INTEGER);

VAR nouv_el : d_el_mc;

BEGIN
   nb_el := 0;
   WHILE (NOT EOF(filin)) AND (nb_el < (n-m+1)) AND (nb_el < 200) DO
      BEGIN
         NEW(nouv_el);
         nouv_el^ := filin^;
         nb_el := nb_el + 1 ;
         tab[nb_el] := nouv_el;
         GET(filin);
      END;
END;
```

```
PROCEDURE write_from_tab(VAR tab : ARRAY[m..n:INTEGER] OF d_el_mc;
                         nb_el : INTEGER;VAR filout : f_el_mc);

VAR i : INTEGER;

BEGIN
   FOR i := 1 TO nb_el DO
      BEGIN
         WRITE(filout,tab[i]^);
         DISPOSE(tab[i]);
      END;
END;


PROCEDURE sort(VAR tab: ARRAY[m..n:INTEGER] OF d_el_mc; nb_el : INTEGER);

VAR i,j,ind_cour : INTEGER;
    l_el_min_cour_content,l_tabj_content : str20;
    d_el_min_cour : d_el_mc;

BEGIN
   FOR i := 1 TO nb_el - 1 DO
      BEGIN
         ind_cour := i;
         d_el_min_cour := tab[i];

         lower(d_el_min_cour^.content,l_el_min_cour_content);

         FOR j := i + 1 TO nb_el DO
            BEGIN
```

```
                lower (tab[j]^.content,l_tabj_content);
                IF l_tabj_content < l_el_min_cour_content THEN
                   BEGIN
                      ind_cour := j;
                      d_el_min_cour := tab[j];
                      lower(d_el_min_cour^.content,l_el_min_cour_content);
                   END;
             END;
          tab[ind_cour] := tab[i];
          tab[i] := d_el_min_cour;
       END;
END; {sort}

{corps de la procedure 'fmc_sort'}
{-------------------------------------------------------------------------}

BEGIN
   RESET(fich);
   read_in_tab(fich,tab_a_trier,nb_el);
   sort(tab_a_trier,nb_el);
   REWRITE(fich);
   write_from_tab(tab_a_trier,nb_el,fich);
END.
```