MASTER EN SCIENCES INFORMATIQUES

Automatisation du dispatching d'un laboratoire d'analyses médicales

Alexandre, Etienne

Award date:
1984

Awarding institution:
Universite de Namur

Institut d'Informatique
21, Rue Grandgagnage
B-5000 NAMUR

ANNEE ACADEMIQUE 1983-84

Automatisation du dispatching d'un laboratoire d'analyses médicales

****

Etienne ALEXANDRE

Promoteur : Mr Jean RAMAEKERS

Mémoire présenté en vue de l'obtention du grade de Licencié et Maitre en Informatique.

Remerciements ...

Je remercie Monsieur J. Ramaekers pour avoir accepté d'être le promoteur de ce mémoire et Monsieur P. Lambion pour m'avoir donné l'occasion d'effectuer un stage très intéressant dans le laboratoire où il est responsable de l'informatique. Je les remercie tous deux pour les conseils qu'ils m'ont prodigués tout au long de ce travail. Enfin, je remercie tout le personnel du laboratoire qui a bien voulu collaborer et procéder aux essais de mes programmes.

Avertissement

Ce mémoire se compose de deux volumes: celui-ci constitue la partie centrale et j'ai essayé de le rendre le plus lisible possible. Il aborde le sujet sans trop donner de détails et en essayant de faire le lien entre les différentes parties. Le deuxième volume contient les annexes: cette partie est nettement moins lisible car elle est constituée des programmes ainsi que d'une série de détails techniques.

Introduction.

1 Cadre général.

Ce mémoire s'est déroulé dans un laboratoire d'analyses médicales et biologiques de Mons. J'ai eu la chance de pouvoir y faire un stage du 15 Août 1983 au 31 Janvier de cette année. J'ai pu, pendant toute cette période découvrir les diverses facettes de la vie de cette entreprise.

Nous sommes en période de crise, et la concurrence entre les laboratoires est féroce. Chacun essaye d'attirer chez lui le plus grand nombre de médecins et de patients c'est-à-dire avoir un trés grand volume d'analyses à effectuer. Comme toute entreprise, elle doit essayer de garder ses anciens clients et d'en conquérir de nouveaux.

Ces clients apprécient la qualité d'un laboratoire en se basant sur trois principaux critères:

1. Rapidité: le temps qui s'écoule entre le moment où le prélèvement est effectué sur le patient et celui où il reçoit ses résultats doit être le plus court possible.

2. Exactitude: les résultats rendus doivent être corrects c'est-à-dire que les méthodes d'analyse doivent être fiables, que les résultats doivent être vérifiés quant à leur cohérence, mais aussi que toute erreur de recopiage (par exemple inversion des résultats de deux patients, décimale mal placée, ...) doit absolument être détectée ou évitée.

3. Présentation des résultats: la qualité de la présentation du document que reçoivent les médecins est assez importante et quand cela est possible, un graphique serait préférable à une série de résultats numériques en vrac.

Ces trois critères justifient pleinement le développement de moyens techniques et technologiques divers:

- Utilisation d'une série de taxis.Ce sont des voitures appartenant au laboratoire qui font leur tournée, pendant la journée pour récolter des prélèvements, le soir, pour porter les résultats. (Critère 1)

- Utilisation de plusieurs machines d'analyse (semi-)automatiques. Certaines peuvent exécuter 12 analyses simultanément sur un même échantillon (machine SMA12) , d'autres font automatiquement en 2 ou 3 heures une analyse qui se faisait manuellement en 2 jours, d'autres encore peuvent effectuer en quelques minutes une analyse sur une trentaine d'échantillons différents (machine COBAS), etc .... (Critères 1 et 2)

- Utilisation de l'informatique pour éditer une série de documents de travail pour les techniciens [1] , pour regrouper les résultats et établir les documents pour les médecins et pour effectuer la facturation. (Critères 1,2 et 3)

Il subsiste actuellement un problème: alors que le laboratoire dispose d'un matériel qui devrait lui permettre de traiter plus rapidement ses analyses, il est freiné par une série de tâches manuelles qui pourraient être automatisées ou informatisées. Ce mémoire s'inscrit dans le cadre de toute une série de mesures et de travaux pour essayer de résoudre ce problème.

2 Objectif du mémoire.

Une de ces étapes lentes et manuelles est le dispatching qui constitue un des éléments vitaux du laboratoire. On peut dès à présent définir le dispatching de laboratoire comme étant l'opération qui consiste à répartir le prélèvement du patient en plusieurs quantités à destination des techniciens. Cette notion sera revue plus en détail.

Actuellement, tout le travail de dispatching se fait manuellement, l'objectif est d'en automatiser la partie la plus routinière et répétitive.

---

1. On appelle ainsi les personnes qui effectuent les analyses.

Chapter 1

Plan du mémoire.

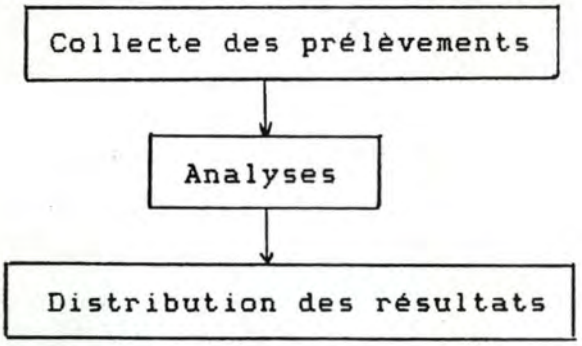
- Description du fonctionnement général du laboratoire.
- Description des différentes étapes de son évolution future.
- Description du dispatching et des problèmes qu'il entraîne: nécessité de données quantitatives (demandes d'analyses) et de données qualitatives (descriptions d'analyses)
- Description de la façon dont sont encodées et gérées les demandes d'analyses (données quantitatives).
- Description de la manière dont est géré le signalétique des analyses (données qualitatives).
- Description et spécifications du programme de dispatching.
- Evocation du problème particulier des COBAS.
- Conclusions.

Chapter 2

Description du laboratoire.

2.1 Fonctionnement général.

Le fonctionnement peut se résumer par un schéma simple :



Les prélèvements à analyser proviennent de plusieurs endroits:

- du laboratoire même où quelques médecins effectuent des prises de sang tous les matins .
- de centres de prélèvements, disposés en ville ou dans la région.
- de médecins, de vétérinaires, ou d'hôpitaux de la région.
- d'autres laboratoires n'effectuant pas certaines analyses.

Les arrivées de prélèvements sont assez groupées car ils sont en général collectés par les taxis. Ces taxis amènent leurs lots de prélèvements entre 8H30 et 16H. Le laboratoire reçoit environ 250 prélèvements par jour, mais en période de pointe, cela peut aller jusque 350.

Le courrier[2] doit être prêt à partir le plus tôt possible (en général pas plus tard que 17H) pour qu'il puisse être emporté par les taxis et arriver le jour même à destination.

En fait, pour que le courrier soit prêt vers 17H, cela implique que les analyses soient terminées bien avant (14H30 au plus tard) afin d'avoir le temps d'effectuer les opérations suivantes:

1. Encoder les résultats des dernières analyses. Les autres sont encodés tout au long de la journée.

2. Charger les derniers résultats dans l'ordinateur.

3. Exécuter le programme d'éditions des protocoles.

4. Imprimer ces protocoles.

5. Les vérifier.

6. Les mettre sous enveloppe.

Pour que les techniciens aient terminé leurs analyses sur un prélèvement pour 14H30, il faut que les demandes relatives à ce prélèvement aient été encodées, que le dispatching ait été effectué et que les techniciens aient reçu leurs listes de travail.

En ayant tenu compte de toutes ces considérations, le laboratoire s'est organisé de la manière suivante: tous les prélèvements qui arrivent avant 11H30 (environ) sont analysés et les résultats sont envoyés le jour même[3] ; ceux qui arrivent après cette heure sont analysés soit en fin d'après-midi soit le lendemain matin et leur résultats sont envoyés avec le courrier du lendemain.

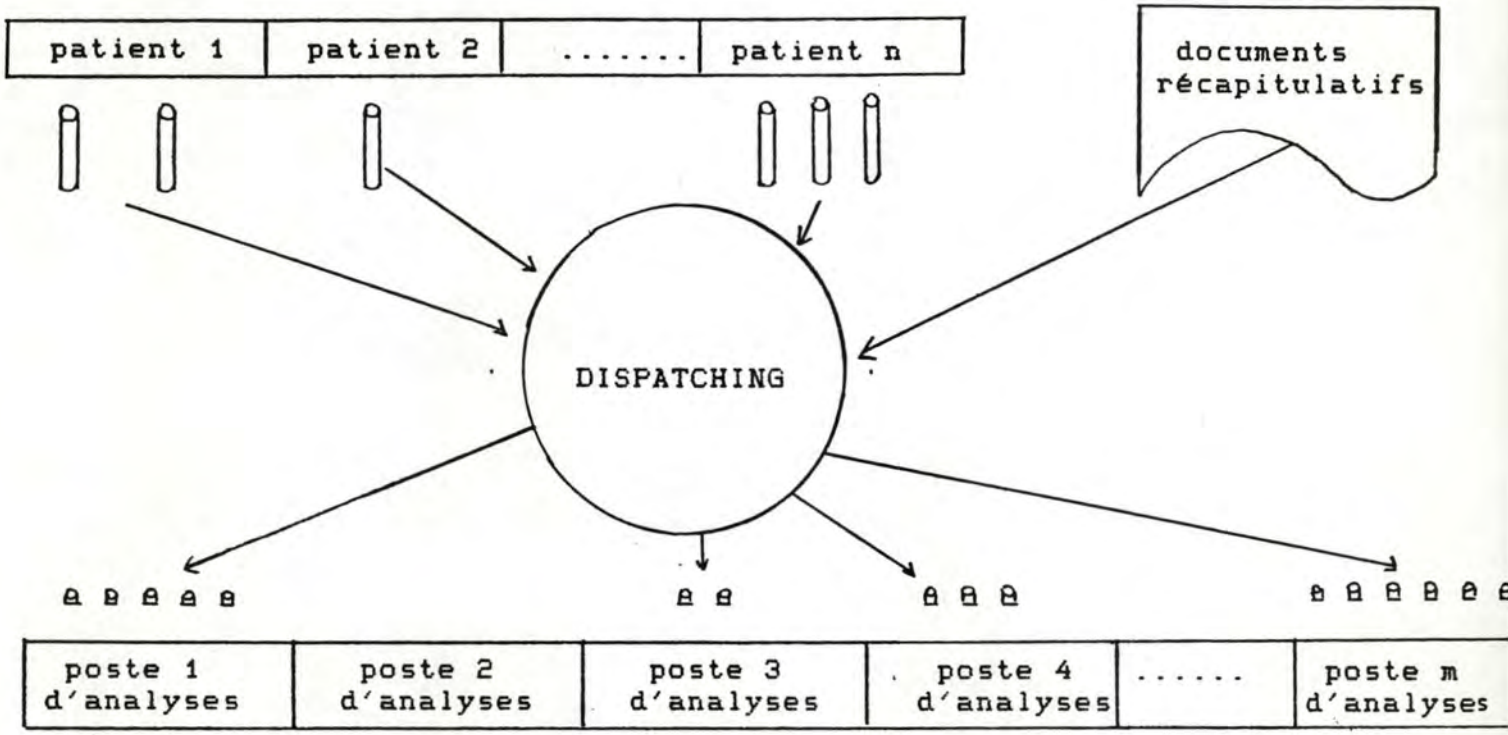
---

2. Ensemble des protocoles (feuilles regroupant les résultats d'un patient) à destination des médecins, vétérinaires, ...

3. Il va de soi que certaines analyses prennent relativement longtemps (incubation, culture,...) et que leurs résultats ne peuvent pas être envoyés avant quelques jours. Il en va de même pour certaines analyses rares, et peu urgentes qui ne sont effectuées qu'une fois tous les x jours.

2.2 Brève "Analyse fonctionnelle" du dispatching.

Schéma:



Le laboratoire reçoit pour un patient

1. Une feuille de demandes d'analyses (Cfr Appendice B).
2. Une ou plusieurs éprouvettes contenant un prélèvement effectué sur ce patient.

Le dispatching d'un patient consiste à fractionner le contenu de ses éprouvettes en une série de petits volumes à destination des différents postes d'analyses.

Cette répartition se fait sur base de documents récapitulatifs créés à partir des demandes d'analyses.

Etant donné qu'après le dispatching, des parties du prélèvement d'un patient peuvent se retrouver à des dizaines d'endroits différents, il est très important d'identifier univoquement tous les échantillons d'un même patient.

La notion de "référence" peut dès lors être précisée: chaque patient se voit attribuer une référence qui servira à l'identifier durant toute la chaîne d'analyse. Cette référence est constituée d'une lettre et de 5 chiffres (ex:J45897, Z12546, N14563). La lettre précise l'origine du prélèvement, le nombre qui suit est attribué par compostage [4]. Cette référence figurera sur toute fraction du prélèvement d'un patient ainsi que sur sa feuille de demandes d'analyses.

2.3 Situation de départ.

Cette situation est celle que j'ai trouvée en arrivant au laboratoire.

2.3.1 Circulation des informations.

Le schéma des flux est représenté à l'appendice A.1.

Dès son entrée au laboratoire, on dispose sur la feuille de demandes (Cfr appendice B) d'analyses ainsi que sur toutes les éprouvettes de prélèvement de ce patient, une étiquette autocollante pré-imprimée par l'ordinateur avec la référence. Cette référence est un moyen très efficace et rapide d'identification de tous les échantillons d'une même personne circulant dans le laboratoire. De plus, cela assure une certaine discrétion puisqu'à aucun moment le technicien ne connaît l'identité de la personne dont provient l'échantillon qu'il manipule.

La feuille de demandes d'analyses arrive au secrétariat. Les secrétaires disposent de deux supports d'encodage:

1. La carte PERFOSTYL. (Cfr appendice C)

2. La feuille de programmation. (Cfr appendice D)

La carte PERFOSTYL est disposée dans un support et recouverte d'une grille en plastique (Cfr appendice C) sur laquelle est imprimée au dessus de chaque position pouvant être trouée, le nom (abrégé) de l'analyse correspondante. La référence a été

---

4. Ceci signifie que chaque numéro, dès qu'il a été utilisé est incrémenté de 1 afin qu'il ne serve pas deux fois

pré-perforée par l'ordinateur. La secrétaire perfore à l'aide d'une pointe en fer tous les trous correspondant aux analyses cochées sur la feuille de demandes. Comme les 800 analyses qu'effectue le laboratoire ne peuvent se retrouver sur la carte PERFOSTYL, les autres doivent être inscrites sur une feuille de programmation. Une fois que les demandes ont été encodées, les feuilles de demandes partent à la salle d'encodage pour l'introduction au terminal des renseignements administratifs. Trois fois par jour, à 9H30, 11H30 et 14H on procède aux opérations suivantes:

- Les feuilles de programmation sont transmises à la salle d'encodage où elles sont perforées sur cartes.

- Les cartes PERFOSTYL sont transformées par l'ordinateur de la banque [5] dont le lecteur de cartes accepte les PERFOSTYL (pas celui du laboratoire) en cartes formatées comme celles sortant de la salle d'encodage. (Crf § précédent)

- Ces deux paquets de cartes de demandes sont réunis et passés dans l'ordinateur du laboratoire. Celui-ci édite les documents suivants:

  * petites listes de travail (Cfr appendice E)

  * tableaux de travail (Cfr appendice F)

  * cartes-réponses (Cfr appendice G)

- Les petites listes et les tableaux sont utilisés pour effectuer le dispatching puis transmis aux différents techniciens avec les échantillons à analyser.

Sur base de leurs documents de travail, les techniciens effectuent les analyses. Suivant les cas,ils peuvent rendre leurs résultats sous deux formes:

1. Une feuille de papier qu'ils ont écrite eux-mêmes ou que la machine d'analyse a imprimée.

2. Des cartes-réponses sur lesquelles sont pré-perforés la référence du patient et le numéro d'analyse. (Une seule analyse et un seul patient par carte) Le technicien y écrit à la main le résultat correspondant à cette analyse pour ce

---

5. La banque et le laboratoire font partie d'un même holding et leurs ordinateurs respectifs se trouvent dans la même salle machine.

patient.

Ces deux sortes de documents sont transmis à la salle d'encodage:

- Les résultats écrits à la main sur les cartes-réponses sont perforés sur ces mêmes cartes, qui deviennent des cartes résultats. (Cfr appendice G)

- Les résultats sur papier sont soit perforés sur carte, soit introduits au terminal.

Tous ces résultats sont fournis à l'ordinateur qui les trie, les regroupe par patient, les formate et enfin édite et imprime les protocoles .

2.3.2 Circulation des prélèvements.

Les prélèvements arrivent directement au dispatching. Ils sont étiquetés et centrifugés en attendant de pouvoir être dispatchés lorsque les documents de travail seront disponibles. Une fois le dispatching effectué, les échantillons partent vers les différents points d'analyse. Ce qui n'a pas été distribué est mis au frigo en réserve.

2.3.3 Dispatching.

2.3.3.1 Distinction.

On distingue deux types de dispatching:

Dispatching immédiat
: Il consiste à effectuer le dispatching du prélèvement dès qu'il arrive au laboratoire pour certaines analyses qui peuvent se faire tout de suite ou qui sont urgentes. On dispatche dans ce cas plusieurs analyses pour un même prélèvement. Ceci en opposition au ...

Dispatching batch
: ... qui consiste à effectuer le même type d'action sur tout un lot de prélèvements pour des analyses qui se font une fois par jour ou même moins souvent. Dans ce cas, on travaille par analyse et on dispatche tous les prélèvements pour lesquels il faut faire cette analyse. Ce type de dispatching s'effectue en fin de journée ou pendant les temps creux.

2.3.3.2 Inputs nécessaires.

Le dispatching a besoin de plusieurs choses pour fonctionner:

- Des échantillons de différentes natures (sérum sanguin, urines, selles,...); ceci constitue sa matière première.

- Des informations à deux niveaux:

  1. A QUI DISTRIBUER ? la réponse à cette question est apportée par les documents de travail ainsi que par quelques indices "matériels": les médecins disposent d'éprouvettes différentes pour certains types d'analyses et ont des consignes pour respecter certaines indications; par exemple, l'analyse X nécessite Y ml de sang dans une éprouvette avec un bouchon rouge.

  2. QUEL VOLUME DISTRIBUER ? ici, il n'y a pas de réponse précise; jusqu'à présent, c'est l'expérience qui a dicté ses lois. Les personnes responsables du dispatching savent plus ou moins que pour telle analyse, il faut tel volume de sérum dans un godet en plastique de telle forme. On travaille donc un peu au hasard et s'il y a beaucoup d'analyses à faire sur un échantillon, le dernier servi risque d'être un peu rationné. Dans ce cas, si le technicien n'a pas assez de sérum pour faire ses analyses, il "suffit" d'aller en rechercher au frigo, s'il en reste. Pendant le même temps, il se pourrait fort bien qu'un autre technicien en aie, lui, beaucoup trop.

- Des récipients de différents types, godets à fond plat, godets à fond conique, éprouvettes en verre, etc ...

- Des instruments de mesure de volumes.

2.3.3.3 Traitement.

Le traitement ne se limite pas toujours à répartir un échantillon entre différents godets, il faut parfois effectuer des dilutions avec certains tampons. Tout se fait manuellement avec toutes les imprécisions et les erreurs que cela peut supposer.

2.3.3.4 Outputs.

Ce qui sort du dispatching est une série de récipients contenant des échantillons à analyser. Ces différents récipients sont identifiés par la référence inscrite dessus.

2.3.4 Critiques du système.

Ce système a été mis en place au début des années 70 et fonctionne très bien depuis cette époque. Toutefois, on peut formuler certaines critiques qui soulignent les faiblesses du système[6]

- Cette solution est fort centralisée; en cas de panne de l'ordinateur central, tout le laboratoire pourrait être bloqué. En effet, le courrier ne sortirait plus et en plus de cela, les techniciens ne pourraient plus travailler puisqu'ils se basent exclusivement sur les documents de travail que l'ordinateur leur fournit. Sans aller aussi loin, une simple panne de deux heures au moment de générer ces documents de travail peut retarder de deux heures tout le processus d'analyse[7]. En fait, cette énorme faiblesse n'a pas souvent posé de graves problèmes; non pas que l'ordinateur central ne soit jamais tombé en panne, mais parce que la banque possède exactement la même machine que le laboratoire et lui sert de backup en cas de panne.

- Il subsiste dans le processus d'analyse trop de tâches manuelles qui pourraient être évitées. Par exemple, beaucoup de machines d'analyses sont équipées de portes de sorties sur lesquelles elles envoient leurs résultats sans qu'aucun dispositif ne les réceptionnne; tout est recopié ou encodé manuellement à partir des listings que sortent également ces machines. Autre exemple, le dispositif d'encodage des demandes est relativement lourd.

---

6. Ces faiblesses ne sont pas dues à une mauvaise conception mais à une technologie encore peu développée à l'époque.

7. Le processus d'analyse est l'ensemble des tâches effectuées depuis le moment où le prélèvement entre au laboratoire et celui où le courrier est prêt à partir.

2.3.5 Proposition d'amélioration.

La principale amélioration proposée, (Cfr Introduction §2) est d'automatiser une partie du dispatching. Comme il s'agit d'une série de manipulations, il faudra faire appel à la robotique.[8] Par chance, le laboratoire dispose déjà d'une machine semblable à celle qui pourrait être utilisée au dispatching. Cette machine est actuellement utilisée en virologie où elle est saturée tant elle rend de grands services. Elle sera détaillée en 9.2 mais il serait bon d'en expliquer dès à présent le fonctionement. Pour une meilleure compréhension, son schéma se trouve en appendice H. Elle est constituée d'une table de travail sur laquelle un bras mobile peut accéder à n'importe quel point avec une précision de 0.1 mm. Ce bras est terminé par une fine aiguille creuse reliée à un système de pompes qui lui permettent d'aspirer ou de refouler des liquides. Les volumes manipulés peuvent varier de quelques microlitres à quelques millilitres. A mon arrivée au laboratoire, le choix de cette machine pour le dispatching était déjà fait.

La deuxième amélioration est de faire de l'encodage des demandes d'analyses un poste de travail autonome bien plus souple que dans la situation de départ. Ceci sera réalisé en effectuant ce travail sur micro-ordinateur. Ici aussi le choix était déjà fait, il s'agit du micro Xérox 820. Ce choix s'est pleinement justifié par la suite. Cette deuxième amélioration s'imposait pour plusieurs raisons:

- Le stock de cartes PERFOSTYL utilisées pour encoder les demandes est presque épuisé et il est très difficile d'en retrouver car ce n'est pour ainsi dire plus employé.

- Le dispatching immédiat, s'il veut être efficace doit être effectué au fur et à mesure que les demandes sont encodées et pour cela le micro qui pilote l'HAMILTON doit être relié en ligne directe avec l'encodage qui dès lors doit être fait aussi sur un micro.

- Etant donné que la banque va changer d'ordinateur, il ne sera plus possible en cas de panne d'aller y générer les documents de travail. Il faut donc que ces documents soient édités par le micro servant à l'encodage.

---

8. MICROLAB 2000 de la firme suisse HAMILTON piloté par un micro-ordinateur de marque KONTRON (Munich).

- Enfin beaucoup de techniciens qui ne recevaient que des petites listes de travail souhaitaient disposer de tableaux qui leur faciliteraient la vie. Or, pour diverses raisons, il est assez difficile d'écrire un programme supplémentaire de création de tableaux sur l'ordinateur central [9] par contre il est assez facile sur un Xérox, d'écrire, en PASCAL ou en C, une procédure paramétrable de génération de tableaux afin de répondre aux désirs émis.

En fait cette solution s'inscrit dans le cadre de la nouvelle configuration préconisée pour le laboratoire, à savoir un réseau de micros reliées à l'ordinateur central.

2.4 Situation de transition.

Cette situation est celle qui sera opérationnelle au terme de ce mémoire.

2.4.1 Circulation des informations.

Le schéma des flux est représenté à l'appendice A.2.

La circulation des informations est à peu près identique à la situation de départ avec les quelques différences suivantes:

- Le secrétariat dispose d'un micro-ordinateur avec trois programmes:

  1. Un programme permettant d'encoder les demandes pour chaque référence.

  2. Un programme qui imprime les tableaux de travail et crée des cartes de demandes qui seront fournies à l'ordinateur central pour obtenir les cartes réponses.

  3. Un programme qui imprime les petites listes de travail.

---

9. Seuls FORTRAN et ASSEMBLEUR sont disponibles.

- Le dispatching devrait [10] disposer également d'un micro pour piloter une HAMILTON qui se chargerait de la partie batch du dispatching.

- Les COBAS [11] ont été reliés à un micro qui organise le travail des techniciens responsables de ces machines à partir des données fournies par le micro de l'encodage des demandes et qui enfin réceptionne les résultats envoyés par ces deux machines et les transforme en cartes-résultats. Ceci permet des écomomies

  * de cartes: là où il fallait 500 cartes pour 500 résultats, maintenant il n'en faut plus qu'une centaine car les résultats sont regroupés par le micro sur les cartes-résultats.

  * de temps: on gagne le temps qu'il fallait à l'ordinateur pour produire les 500 cartes-réponses, le temps que mettaient les techniciens à recopier les résultats sur les cartes-réponses et enfin, le temps mis par les encodeuses pour perforer ces résultats.

2.4.2 Circulation des prélèvements.

Elle est identique à la situation de départ mais le dispatching devrait se faire partiellement par l'HAMILTON ce qui devrait accélérer la circulation des échantillons.

2.4.3 Dispatching.

Comme il a été dit plus haut, une partie devrait être automatisée mais l'opérationnalité de cette partie est retardée jusqu'à l'arrivée de la machine. Le dispatching immédiat continue à se faire manuellement.

---

10. En fait pour des raisons budgétaires, L'HAMILTON (1.750.000 frs + le micro) n'a pas encore été achetée et le dispatching se fait encore manuellement malgré que le programme soit écrit et testé.

11. Ce sont deux machines d'analyse automatiques qui effectuent par jour environ 500 analyses et dont tous les résultats sont retranscrits sur des cartes-réponses.(Ces cartes-réponses pour les COBAS constituent la plus grosse part de toutes les cartes-réponses utilisées sur une journée.)

2.4.4 Critiques du système.

Comme dans la plupart des projets informatiques, ce sont les problèmes d'organisation qui ont été déterminants. Il a fallu que les secrétaires se familiarisent au fonctionnement et à la manipulation d'un micro-ordinateur et de ses périphériques, ce qui n'a pas été sans mal puisqu'elle n'avaient jamais été en contact avec ce genre de machines.

Le système d'encodage présente des avantages non négligeables

- corrections très faciles
- consultations rapides
- autonomie par rapport à l'ordinateur central
- souplesse
- création immédiate de fichiers de demandes pouvant être utilisés par les autres micros disséminés dans le laboratoire
- etc...

Il présente toutefois un inconvénient par rapport à la situation de départ: la lenteur. Dans l'ancien système, lorsqu'il y avait une arrivée massive de prélèvements, les secrétaires pouvaient se mettre à perforer leur cartes à 2 ou 3 tandis que maintenant, il n'y a qu'un clavier et une seule personne peut encoder à la fois.

2.4.5 Proposition d'amélioration.

Les quelques propositions suivantes devraient permettre d'arriver à la situation finale:

- Ajoute d'un deuxième micro à l'encodage afin d'absorber les pointes.
- Encodage des renseignements administratifs sur micro.
- Disposition de plusieurs micros dans le laboratoire aux différents postes de travail pour permettre certains travaux

de calcul ou autres[12] en plus de l'introduction des résultats d'analyse.

- Remplacement de l'ordinateur central afin de pouvoir supporter des liaisons directes avec les micros, ce qui n'est actuellement pas le cas: le seul interface entre les micros et le central est la carte perforée. Le choix de la nouvelle machine est déjà pris; l'IBM 370-125 va être remplacé par un MV 4000 (Data General). Les deux machines tourneront en parallèle à partir de fin Septembre 1984 et ce jusque Février-Mars 1985, date à laquelle l'IBM sera définitivement arrêté. Cette période de transition devrait permettre la réécriture[13] des programmes et leur test tout en tournant toujours avec l'ancien système pour des raisons de sécurité.

- Etablissement de lignes directes entre les divers micros et l'ordinateur central.

2.5 Situation finale.

Cette situation est celle qui deviendra progresivement opérationnelle à partir de Mars-Avril 1985. Cette situation est encore un peu floue et je n'en donnerai que les grandes lignes.

2.5.1 Circulation des informations.

Le schéma des flux est représenté à l'appendice A.3.

Toutes (ou presque)[14] les informations circuleront par ligne et non plus par carte ou papier.

---

12. par ligne ou manuellement suivant les machines utilisées à ces postes

13. L'incompatiblité entre les deux machines est totale.

14. Etant donné qu'il peut présenter des avantages dans certains cas, le système de cartes pourrait survivre par endroits mais dans des proportions négligeables

2.5.2 Circulation des prélèvements.

Elle ne changera pas beaucoup par rapport aux deux situations précédentes mais le dispatching immédiat devrait être opérationnel.

2.5.3 Dispatching.

Les dispatchings immédiat et batch devraient être opérationnels. Ceci ne signifie nullement que tout se fera automatiquement: il restera toujours une partie des traitements qui ne seront pas automatisés. (Ex: urgences, manipulations de corps non liquides ou de gros volumes (quelques cl) , etc...) Il faut signaler que la rapidité des personnes assurant le dispatching n'est pas ici mise en cause; le problème est qu'elles sont souvent surchargées.

Le fait d'automatiser leur tâche ne va pas l'accélérer, bien au contraire; la dextérité et la rapidité de ces personnes n'ont rien à envier à une HAMILTON, mais une telle machine pourrait rendre de grands services en les déchargeant d'une partie de leur travail. Le travail ainsi fourni en parallèle par la machine d'un côté et les personnes de l'autre pourrait alors être globalement amélioré et accéléré.

2.5.4 Critiques du système.

Le système va présenter tous les avantages d'une configuration décentralisée (si l'ordinateur central tombe momentanément en panne, les micros peuvent conserver leurs résultats) et d'un matériel standard (un des micros servant au développement sera considéré comme "réserve" et pourra remplacer endéans les quelques minutes n'importe quel micro du laboratoire qui serait défaillant.

Pour ce qui est des critiques négatives, il est difficile de se prononcer maintenant: "Wait and see".

Chapter 3

Problématique du dispatching.

3.1 Au niveau logiciel.

Pour pouvoir être automatisé, le dispatching doit disposer de toute une série d'informations:

- fichier des demandes pour savoir quels prélèvements dispatcher (Cfr 4.2.3)
- fichiers de description des analyses pour savoir quels volumes dispatcher et éventuellement quels tampons utiliser (Cfr 5.3)
- plus encore différents autres fichiers pour disposer d'informations nécessaires (Cfr 6.2)

Une partie de ces fichiers est créé sur le micro de l'encodage, les autres doivent être crées et gérés par le micro qui pilote l'HAMILTON. Ce même micro devra, dans le cas du dispatching immédiat, réceptionner les demandes au fur et à mesure qu'elles sont encodées afin de pouvoir effectuer le dispatching immédiatement. Cette réception devra se faire par ligne.

A première vue, tout ceci ne devrait poser aucun problème majeur, mais comme nous allons le voir, ce n'est pas le cas.

3.2 Au niveau matériel.

Comme cela a été dit plus haut, c'est le MICROLAB 2000 de chez HAMILTON qui a été choisi pour effectuer automatiquement le dispatching. Cette machine est pilotée par un micro d'une firme munichoise assez peu connue : KONTRON. Comme cela a été dit également, le choix des micros pour le réseau du laboratoire s'est porté sur les Xérox 820.

C'est ici que les problèmes commencent; les Xérox et le KONTRON ne sont absolument pas compatibles. Les Xérox, tournent sous CP/M avec des disquettes 8 pouces, le KONTRON, lui, tourne sous KOS avec des disquettes 5 pouces. De plus, il n'est programmable qu'en BASIC. Tout l'utilitaire de pilotage de l'HAMILTON est d'ailleurs écrit pour le BASIC. Il est évident que s'il faut se mettre à programmer le dispatching en BASIC en utilisant toute une série de fichiers et en plus en gérant une deuxième porte[15] de sortie pour receptionner les demandes, cela risque de poser certains problèmes.

Ils apparaît dès lors très intéressant de pouvoir utiliser un Xérox et non plus un KONTRON pour le pilotage et ceci se justifie par les raisons suivantes:

- La standardisation est respectée, et en cas de panne on peut remplacer le Xérox de pilotage par un autre identique, ce qui n'est pas le cas avec un KONTRON puisque le laboratoire n'en possède qu'un.

- A l'achat, le KONTRON coûte environ 150.000 frs plus cher que le Xérox. Ceci pourrait représenter une sérieuse économie lors de l'achat d'une HAMILTON pour le dispatching.

- Les programmes de pilotage pourraient se faire en PASCAL ce qui est une nette amélioration par rapport au BASIC.

- Les communications avec un autre Xérox seraient bien plus faciles à partir d'un Xérox qu'à partir d'un KONTRON.

Toutefois, cette substitution ne s'avère pas évidente, en effet:

- le gérant des communications entre le KONTRON et l'HAMILTON est écrit en assembleur et le laboratoire ne dispose que de la version exécutable; le constructeur ne ne désire pas fournir la version source.

- Aucune documentation n'est fournie sur les caractéristiques du protocole de transmission (vitesse, parité, ...)

Le pilotage apparaît donc comme une boîte noire dans laquelle on fait entrer des ordres qu'elle se charge de faire exécuter par le robot (HAMILTON). Parfois, cette boîte renvoie un message signalant une erreur de tel ou tel type. C'est le mystère de

---

15. La première servant au pilotage de l'HAMILTON.

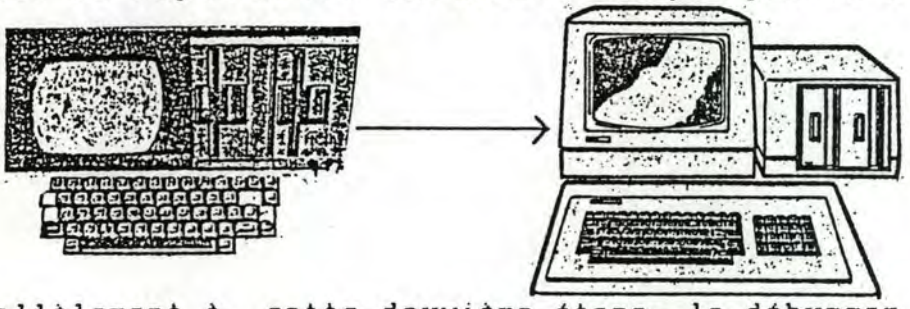
cette boîte noire qu'il s'agit de percer. On connaît très précisement les ordres qui y rentrent, mais on a aucune idée sur la manière dont ils sont transmis et/ou transformés avant cet envoi.

3.3 Analyse de la boîte noire.

Voici les différentes étapes de cette analyse:

1. La consultation de toute la documentation fournie avec le KONTRON a donné pour seuls résultats quelques renseignements sur le cablage des fils utilisés entre les deux appareils.

2. Dans une deuxième étape, le KONTRON a été relié à un Xérox pour essayer de trouver les bonnes caractéristiques physiques à utiliser (vitesse, ...) et pour essayer de capter la séquence des caractères envoyés par le KONTRON.



Parallèlement à cette deuxième étape, le débugger a permis d'avoir une version plus ou moins lisible du gérant des communications.

C'est en essayant de déchiffrer ce programme, qu'est apparue la partie initialisation qui a donné les caractéristiques physiques de la ligne [16]. Cette découverte a permis de faire avancer les tests mais il n'a pas été possible tout de suite de faire envoyer par le KONTRON toute une série de commandes comme s'il s'adressait au robot car il "buttait" et lancait quelques bips sonores après avoir envoyé sa première commande.

Ceci a confirmé l'hypothèse que la boîte noire correspondait en fait à tout un dialogue entre les deux

---

16. 2400 bauds, 8 bits/car. ,pas de parité, 2 stop bits

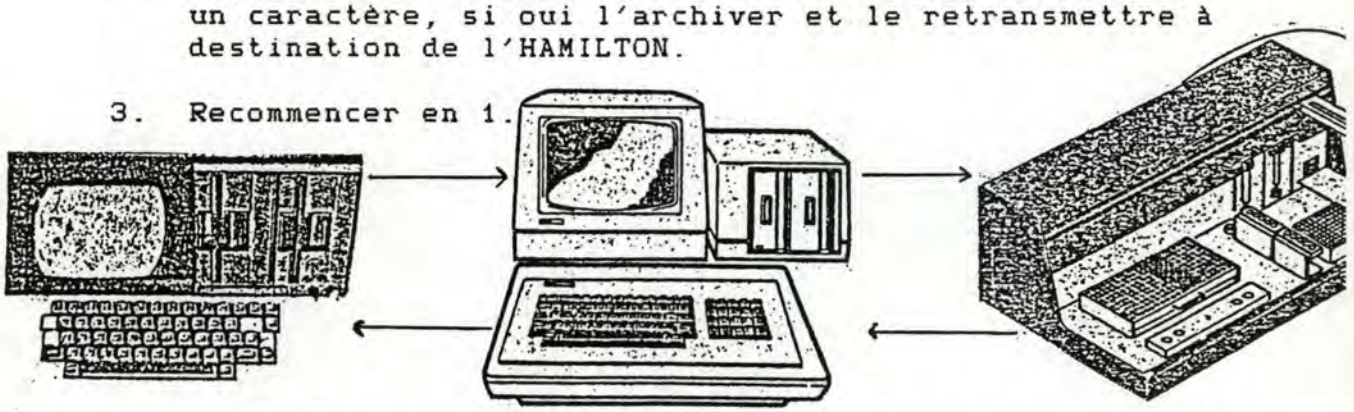
machines mais il n'était pas encore possible à ce niveau des recherches de simuler les réponses de l'HAMILTON par le Xérox pour la bonne raison qu'il n'y avait absolument aucune information sur ce que pouvait bien répondre l'HAMILTON aux ordres du KONTRON.

3. La troisième étape était destinée à espionner ce dialogue. Cela a été réalisé en mettant le Xérox comme espion sur la ligne. Le système est très simple, on a coupé le cable reliant le KONTRON à l'HAMILTON, et on a relié ces deux bouts aux deux portes de sortie du Xérox. Un petit programme tout simple réalisait les opérations suivantes:

    1. Regarder sur la porte reliée à l'HAMILTON s'il a envoyé un caractère, si oui l'archiver et le retransmettre à destination du KONTRON.

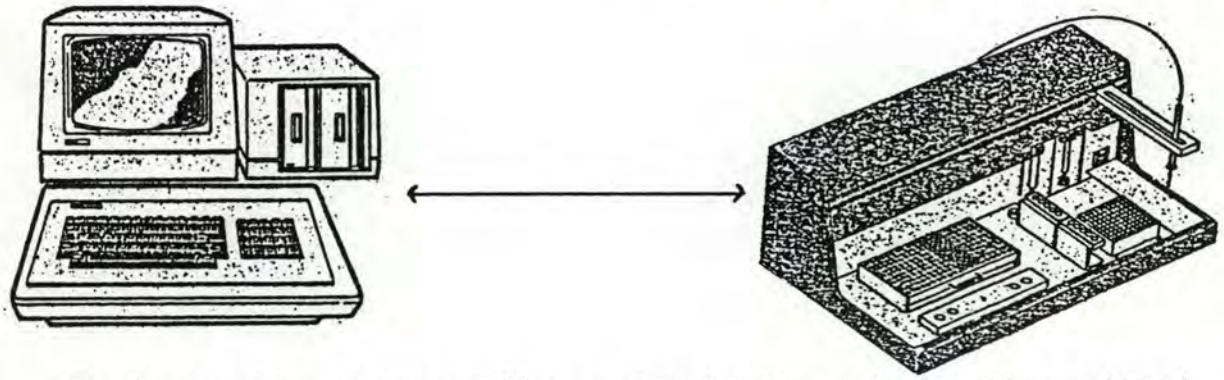
    2. Regarder sur la porte reliée au KONTRON s'il a envoyé un caractère, si oui l'archiver et le retransmettre à destination de l'HAMILTON.

    3. Recommencer en 1.



L'observation de tout ce qui avait été archivé a permis de découvrir les mécanismes et la syntaxe de ce dialogue. Toujours parrallèlement à cette étape, l'"épluchage" du gérant des communications s'est poursuivi et les choses se sont éclairées peu à peu grâce à des recoupements entre quelques fonctions de ce gérant et les tests effectués.

4. La dernière étape visait à piloter l'HAMILTON directement à partir du Xérox. Il aurait théoriquement suffi de reproduire des dialogues correspondant à la syntaxe dégagée de l'étape précédente pour que cela marche mais cela n'en a pas été ainsi.



Pendant des jours entiers, les tests n'ont rien donné; l'HAMILTON ne daignait pas bouger. Ce n'est que grâce à un énorme coup de chance que le mystère a pu être percé: comme l'HAMILTON n'avait aucune réaction aux ordres que le Xérox lui transmettait, les programmes ont été vérifiés et revérifiés mais en vain; pour finir, afin de vérifier ce qui était réellement transmis, une ligne de programme a été rajoutée pour que les caractères transmis apparaissent sur l'écran et tout d'un coup, ça a marché, l'HAMILTON obéissait et répondait. Toutefois, quand cette ligne était supprimée, ça ne marchait de nouveau plus.

Finalement, la lumière s'est faite, il fallait un léger délai entre l'envoi de chaque caractère. Il a donc suffi de rajouter une boucle de temporisation entre chaque envoi, et cela a marché.

3.4 Réécriture de la boîte noire.

Tous les secrets de cette boîte noire étant percés, il fallait encore la réécrire pour qu'elle puisse fonctionner sur un Xérox. Un gérant des communications ainsi qu'une série de procédures facilitant grandement le pilotage de l'HAMILTON ont donc été réécrits en PASCAL

3.5 Réécriture des programmes existants.

L'HAMILTON est utilisée en virologie où elle exécute pendant toute la journée deux programmes qui rendent d'énormes services. L'occasion était belle de tester si la boîte noire qui avait été réécrite était pleinement opérationnelle et les programmes BASIC

du KONTRON ont été recopiés [17] en PASCAL sur Xérox.

Les essais ont été concluants, le KONTRON pourrait être complètement éliminé au profit d'un Xérox.

3.6 Remarque.

Tous les détails techniques ainsi que le texte des programmes, se trouvent dans les annexes. Toutefois, les deux programmes dont il a été question à la section précédente ne sont pas publiés pour deux raisons:

1. Ils sont totalement confidentiels.

2. Le PASCAL ou plutot le BASIC PASCALisé utilisé pour les réécrire ferait honte à n'importe quel programmeur sérieux et comme il ne s'agit que de programmes annexes, ils ne figurent pas dans ce mémoire.

---

17. C'est davantage du recopiage qu'une réécriture car par manque de temps, ces programmes (par ailleurs très compliqués) n'ont pas été analysés; ils ont simplement été recopiés de BASIC en PASCAL en y changeant le minimum pour qu'il soient acceptés par le compilateur PASCAL et en utilisant les procédures qui avaient été écrites auparavant (Cfr section précédente)

Chapter 4

Gestion des demandes.

4.1 Introduction.

Il s'agit ici de répondre à la question

"A QUI DISTRIBUER ?" (Cfr 2.3.3.2)

Comme cela a été dit en 3.1, le fonctionnement du dispatching nécessite l'existence d'un fichier des demandes.

Ces demandes vont devoir être encodées, éventuellement corrigées et des documents récapitulatifs devront être édités. Ces programmes doivent être rapides, souples et "user friendly". C'est ce que ce chapitre se propose de présenter. Il s'agit d'une présentation générale, les programmes seront expliqués en détail au chapitre 10.

4.2 Encodage des demandes.

4.2.1 Objectifs.

Au départ, le programme d'encodage des demandes devait uniquement servir à créer des fichiers de test pour le dispatching mais comme le besoin de faire du poste d'encodage un poste autonome pour les raisons expliquées en 2.3.4 ce programme a du être développé au maximum et est devenu le plus volumineux de ce mémoire.

4.2.2 Principe.

Le principe général en est assez simple:

- l'écran présente la prochaine référence dont il faut encoder les demandes; rappelons que le numéro de référence pour une lettre donnée est attribué par l'ordinateur.(Cfr 2.2)

- le choix de la lettre de la référence se fait simplement en appuyant sur la touche correspondant à la lettre désirée[18] et l'ordinateur affiche la prochaine référence commencant par cette lettre. Exemple: J15478 est affiché, si on tape Z, il va afficher le prochain Z à encoder, par exemple Z12546.

- on signale par une touche spéciale qu'on veut commencer à introduire les demandes pour la référence affichée, puis on tape les numéros des analyses figurant sur la feuille de demandes.

  Lorsque cela est terminé, on presse à nouveau une touche spéciale; tout ce qui a été tapé est enregistré, l'écran est nettoyé et affiche la prochaine référence à encoder.

  Par exemple, si on vient d'introduire les demandes pour la J23651, une fois ces demandes enregistrées, l'écran va afficher J23652 et on pourra dès lors introduire les demandes pour cette référence ou sélectionner une autre lettre.

  Ce système a l'énorme avantage d'obliger les secrétaires à encoder les références en suivant l'ordre donné par l'ordinateur, c'est-à-dire sans en oublier puisque les feuilles de demandes ont reçu dès leur entrée au laboratoire une référence grâce à une étiquette autocollante. (Cfr 2.3.1)

  Exemple: prenons une feuille de demandes qui arrive au laboratoire; la secrétaire y colle une étiquette de la prochaine ligne d'étiquettes[19] disponible (ex: N45684),

---

18. Actuellement, on encode uniquement des référence commençant par les lettres suivantes: B,J,L,N,V,Z

19. Elle dispose en fait pour chaque référence d'une ligne d'étiquettes (8 par ligne) identiques puisqu'il en faut une pour la feuille de demandes et une par éprouvette de prélèvement. Une ligne entamée pour un patient est ensuite barrée puisqu'elle ne peut plus servir pour un autre. (Deux patients différents ou le même patient qui revient le lendemain ont <u>toujours</u> des références différentes.)

ainsi que sur les prélèvements arrivés avec cette feuille. Au moment donc où l'ordinateur affichera N45684, celle qui encode ne pourra rien faire d'autre que d'introduire les demandes de la feuille marquée de cette référence.

- à chaque fois qu'il enregistre les demandes pour une référence, l'ordinateur envoie sur imprimante cette référence ainsi que la listes des analyses demandées (numéro + libellé) pour celle-ci.(Cfr appendice L)

  Ce listing servira à vérifier l'encodage en comparant la feuille de demandes et le listing. Les corrections sont effectuées grâce au même programme que celui qui a servi à encoder; au lieu de dire qu'on veut commencer à introduire les demandes pour la référence affichée, on peut demander d'aller rechercher une telle référence afin de la corriger, c'est-à-dire retirer des analyses en trop ou en rajouter des manquantes.

Les fonctions moins importantes de ce programme ainsi que le programme lui-même seront détaillés au chapitre 10.

4.2.3 Structure des donneées.

Le schéma de cette structure se trouve en appendice I.

Les deux principaux fichiers issus de l'encodage des demandes sont:

FILREF — Fichier des références; il contient les différentes références encodées.

FILDDE — Fichier des demandes; il contient pour chaque référence de FILREF un tableau de 1024 bits dans lequel sont enregistrées les demandes.

Il existe une bijection entre ces deux fichiers, c'est-à-dire que la ième référence de FILREF correspond uniquement au ième tableau de bits de FILDDE et vice versa.

4.3 Documents de travail.

4.3.1 Objectifs.

Le laboratoire compte une vingtaine de techniciens qui sont chargés d'analyses différentes. Il faut donc qu'à un moment donné, chacun dispose

- des échantillons à analyser
- de la liste des analyses à effectuer sur ces échantillons

On pourrait donc envisager de faire de chaque feuille de demandes une photocopie en 20 exemplaires que l'on distribuerait aux différents techniciens. Ceci n'est guère envisageable pour les raisons suivantes:

- Cela ferait environ 500 feuilles [20] par technicien soit 10000 photocopies par jour donc extrèmement coûteux.
- Chaque technicien, au moment de faire l'analyse Y devrait parcourir ses 500 feuilles pour voir sur lesquelles est cochée Y donc énorme perte de temps.
- Le dispatching devrait faire les mêmes manipulations pour préparer ses godets pour chaque analyse.

Cette solution est absolument proscrite et l'idée a été retenue depuis l'informatisation du laboratoire de faire faire par l'ordinateur un travail de regroupement et d'édition. Dans la situation de départ, ce travail est effectué par l'ordinateur central, dans les solutions suivantes, ce travail sera fait par le Xérox servant à l'encodage des demandes. Des documents de deux types sont produits et expliqués ci-après.

4.3.2 Petites listes de travail.

Ceci est le premier type de document auquel on pense tout de suite, il donne par analyse la liste des références à analyser. (Cfr appendice E)

Le programme réalisant ceci est assez simple mais son implémentation l'est moins. Rappelons deux chiffres: environ 250 patients par jour (pointes de 350 à envisager), 800 analyses différentes possibles.

---

20. +/- 250 feuilles par jour, recto verso

Il faut également signaler que pour diverses raisons, les demandes sont mémorisées d'une manière assez spéciale: pour chaque référence, on dispose d'un tableau de 1024 bits avec la convention que l'analyse n° i est demandée si et seulement si le iéme bit du tableau est à 1. Lorsqu'on désire savoir quelles analyses sont demandées pour une référence, il n'y a pas d'autres solutions que de passer en revue tous les bits [21]. Comme on compte en moyenne environ 20 analyses demandées par patient, cela fait 780 tests négatifs et inutiles mais néanmoins obligatoires.

Le programme pourrait donc procéder de deux manières différentes.

1. Procéder par analyse: pour chacune des 800 analyses lire le tableau de bits de tous les patients l'un après l'autre en testant chaque fois le bit correspondant à l'analyse concernée [22] c'est-à-dire environ 200000 [23] accès à la disquette. Cette solution est irréalisable sur micro étant donné le temps d'accès à la disquette.

2. Procéder par patient: pour chaque patient, lire le tableau de bits, et y tester les 800 analyses. Ceci ne nécessite que 250 accès disquette mais il faudrait 800 tables (1 pour chaque analyse) pouvant contenir environ 200 références (estimation grossière) pour mémoriser, pour chaque analyse, la liste des références à analyser. Ceci nécessiterait 960 KB [24] en mémoire pour ces tables. On pourrait essayer de remédier à ce manque de place mémoire en utilisant des fichiers mais cela prendrait de nouveau plus de temps en accès disquette sans parler du temps nécessaire pour effectuer les 200000 [25] tests pour voir quelles analyses sont demandées pour chaque patient. Cette solution semble

---

21. Seuls les 800 premiers bits sont testés car actuellement les numéros des analyses effectuées au laboratoire ne vont que jusque 800

22. 1 accès disquette par patient car on ne peut les charger tous en mémoire

23. 800 analyses * 250 patients

24. 200 références * 6 caractères par référence * 800 analyses

25. 250 patients * 800 analyses possibles

donc également irréalisable.

Il a donc fallu procéder autrement; la solution adoptée évite trop d'accès disquette d'une part et les 800 tests d'autre part.

Il suffit d'établir ces listes au moment de l'encodage, de la manière suivante:

- au moment d'enregistrer les demandes qui viennent d'être introduites pour une référence, le programme d'encodage dispose en mémoire d'un tableau contenant les numéros des analyses demandées. C'est à cet endroit que les demandes sont transformées (pour leur mémorisation) en une série de bits à 1.

  Exemple: si la table contient les numéros d'analyse suivants: 17, 33, 48, ..., 568 le programme va positionner les 17ème, 33ème, 48ème, ..., 568ème bits d'un tableau qu'il va enregistrer pour la référence concernée. Pour notre problème de listes, supposons qu'il y ait 34 analyses demandées, il faudrait donc aller ranger la référence dans les 34 listes correspondant aux 34 numéros d'analyse, ce qui nous ramène au problème de la place nécessaire pour ces tables. Une méthode utilisant nettement moins de place a donc été utilisée. Pour chacun des 34 numéros d'analyse dont on dispose, on écrit sur fichier un couple (numéro d'analyse,référence).

  Exemple: Si les 34 numéros concernent la référence J22333, alors on va écrire dans le fichier

  17J22333
  33J22333
  48J22333
  ..
  ..
  568J22333

  Lorsqu'on a fait ceci pour toutes les références au fur et à mesure qu'elles sont encodées, le programme de création des petites listes de travail n'a plus qu'à charger le fichier ainsi constitué et le trier. Ce tri va donc donner une liste du genre:

  1J45897
  1J45900
  1N21547
  1Z14589
  1Z14590
  2J45899
  2J45900

10Z14590
10Z14591
..
..
..

qu 'il suffira d'imprimer avec un système de rupture sur le numéro d'analyse pour obtenir les listes désirées à peu de frais.

Signalons enfin que seules les analyses qui ne sont pas traitées par les tableaux dont il est question au point suivant sont reprises dans ces listes.

4.3.3 Tableaux de travail.

L'idée est toujours ici de savoir sur quels échantillons effectuer telle analyse, mais pour des raisons de facilité, certains techniciens souhaiteraient disposer de tableaux plutôt que de petites listes. Des exemples des différents tableaux se trouvent en appendice F.

La solution adoptée ici ressemble un peu à la deuxième solution proposée pour les petites listes en y éliminant les inconvénients: On va charger l'un après l'autre les clients mais ici, on ne devra plus tester les 800 analyses; en effet pour chaque tableau, on connait les numéros des analyses à tester c'est-à-dire qu'au total de tous les tableaux, dans l'état actuel du programme, on va tester 142 analyses.

Le problème de la mémorisation est également résolu. Il est assez difficile de dire que pour tel type de tableau il faudra au maximum n lignes en mémoire car cela dépend fortement du nombre de références traitées, ainsi que de l'origine des prélèvements [26] ; ou alors, il faudrait choisir des chiffres tellement grands que la somme de ces chiffres pour tous les tableaux dépasserait la capacité mémoire.

On a donc réservé un certain nombre de lignes globalement pour tous les tableaux et le problème a été réglé. Toutes ces lignes possèdent un code qui permet de savoir à quel tableau elles appartiennent et un tri sur ce code permet de regrouper les lignes par tableau et de les imprimer.

---

26. Suivant l'origine,(vétérinaires, autres laboratoires, ...) c'est plutôt telles analyses qui seront demandées plutôt que telles autres

4.4 Programmes de récupération.

4.4.1 Problèmes à résoudre.

Il y deux problèmes assez importants qu'on ne peut pas empêcher mais dont il faut réparer les dégâts.

1. Le bourrage lors de l'impression
2. La panne de courant

4.4.2 Récupération en cas de bourrage.

Les problèmes durant l'impression des tableaux sont très rares; les pages sont bien remplies, l'impression se fait assez lentement et les sauts de pages sont peu nombreux. Le programme SOS cerne le problème par une série de questions précises et en cas de problème, il permet éventuellement de restaurer de manière interactive les bonnes valeurs dans différents fichiers utilisés par le programme de création des tableaux avant de relancer celui-ci.

L'impression des petites listes de travail pose, elle, plus de problèmes: les lignes sont presque vides (en général une référence par ligne), on n'utilise qu'une ligne sur deux, les pages ne sont pas souvent utilisées jusqu'au bout et les sauts de page sont très nombreux; tout ceci implique une impression assez rapide et sujette à des bourrages. En cas de problème, le programme SOS permet de restaurer le fichier utilisé [27] et de recommencer le programme d'impression.

4.4.3 Récupération en cas de panne de courant.

Ce programme est devenu nécessaire à cause d'une particularité de CP/M: toute extension d'un fichier est perdue si le fichier n'est pas fermé à la fin du programme. Ceci veut dire que lors d'une panne de courant, on ne passe pas par la fin du programme

---

27. Il s'agit du fichier évoqué en 4.3.2. Après l'impression il change de nom pour servir de backup et pour que les prochaines références encodées ne soient pas cumulées dedans.

et tout ce qui a été introduit dans certains fichiers depuis la dernière fermeture est perdu. Certains fichiers auxquels on accède d'une manière particulière sont même détruits. Pour des raisons qui seront évoquées plus loin, de telles pannes détériorent gravement la cohérence entre certains fichiers. Pour ces raisons, le programme d'encodage ferme puis rouvre les fichiers chaque fois que 16 références ont été encodées et le programme PANNE permet de rétablir la cohérence des fichiers.

Chapter 5

Gestion des analyses.

5.1 Introduction.

Il s'agit ici de répondre à la question

"QUEL VOLUME DISTRIBUER ?" (Cfr 2.3.3.2)

Comme je l'ai déjà dit, il n'y a aucune information précise sur les volumes à dispatcher et tout se fait un peu au hasard. Il semble donc important de remettre un peu d'ordre dans tous cela, d'autant plus que le programme de dispatching devra avoir, lui, des données précises pour travailler.

Il va donc falloir créer et gérer un signalétique dans lequel, pour chaque type d'analyse, on va enregistrer divers renseignements parmi lesquels figurera celui qui nous intéresse ici: le volume de prélèvement nécessaire pour effectuer une analyse de ce type. Ce volume sera établi d'après les desiderata précis des techniciens effectuant les analyses.

Cette partie a donc pour but de gérer la liste des types d' analyses effectuées au laboratoire ainsi que leur description.

5.2 Informations relatives à une analyse.

Chaque analyse est décrite par les informations suivantes:

NUMERO — Numéro identifiant chaque analyse au niveau secrétariat et médecins.Ce numéro n'a rien a voir avec le numéro inami.

LIBELLE — Libellé en 25 caractères maximum de l'analyse.

PRELEVEMENT — Type de prélèvement sur lequel va s'effectuer l'analyse. Les différents prélèvements utilisés sont les suivants:

- SANG
- URINE
- SELLES
- LIQUIDE GASTRIQUE
- LIQUIDE AMNIOTIQUE
- PONCTION
- L.C.R. (liquide céphalo-rachidien)
- DIVERS

QTE DE PRELEVEMENT — Quantité de prélèvement utilisée pour cette analyse.

TECHNIQUE — Technique utilisée pour l'analyse:

- CHIMIE
- ISOTOPES
- DIVERS

DISPATCHING — Mode de dispatching: BATCH ou IMMEDIAT.

DOUBLE RINCAGE — Oui ou non la manipulation de ce prélèvement nécessite-t-elle un double rincage.

NUMERO1 TAMPON — Numéro du tampon utilisé si on utilise le prélèvement en dilution.

QTE1 TAMPON — Quantité utilisée de ce tampon.

NUMERO2 TAMPON — Numéro d'un éventuel deuxième tampon utilisé.

QTE2 TAMPON — Quantité utilisée de ce tampon.

DATE — Dernière date à laquelle a été modifiée cette analyse.

COMMENTAIRE — Commentaire éventuel en 70 caratères maximum.

LISTE — Liste des analyses pouvant avoir exactement les mêmes caractéristiques (excepté NUMERO et LIBELLE) que celle-ci.

5.3 Structure des données.

Cette structure est reprise dans le schéma de l'appendice J.

Pour diverses raisons qui seront expliquées plus bas, toutes ces informations ont été réparties dans 3 fichiers distincts.

1. FILANA: fichier des analyses; chaque article contient:
   - Le numéro d'analyse.
   - Le libellé.
   - Le numéro de la description spécifiant les caractéristiques de cette analyse.
2. FILDSC: fichier des descriptions d'analyse; chaque article contient:
   - Le numéro de cette description.
   - Le type de prélèvement.
   - La quantité utilisée.
   - La technique.
   - Le mode de dispatching.
   - L'option double rinçage.
   - Le numéro et la quantité des tampons utilisés.(maximum 2)
3. FILCOM: fichier des commentaires; chaque article contient:
   - La date.
   - Le commentaire en 70 caractères.
   - La liste des analyses pouvant être décrites par la description à laquelle est associé cet article.

Les informations contenues dans FILCOM ne sont pas utilisées par le dispatching et c'est la raison pour laquelle elles font l'objet d'un fichier séparé.Il est à remarquer qu'il existe une relation de bijection [28] entre les fichiers FILCOM et FILDSC.

D'autre part, il est très possible que différentes analyses puissent avoir les mêmes caractéristiques; il est donc avantageux de n'avoir pour celles-ci qu'une seule description commune. Exemple: si les analyses 17, 56, 89 ont les mêmes caractéristiques, on va créer une description (ex: N° 8) qu'on va leur associer à toutes trois. Ceci sera tout simplement réalisé en attribuant la valeur 8 au champ "NUMERO DE DESCRIPTION" de ces trois analyses.

Le champ "LISTE" se trouvant dans FILCOM, est très utile en effet, supposons qu'une caractéristique de l'analyse 17 a changé, il faut donc aller modifier sa description(la N°8). Si cette description n'est utilisée que pour l'analyse 17, il n'y a pas de problème mais si comme c'est le cas ici, elle est également utilisée pour décrire les analyses 56 et 89, une modification de la description pour l'analyse 17 va également modifier la description des analyses 56 et 89 ce qui n'est pas nécessairement voulu.

Il y a donc deux cas à distinguer:

1. Les caractéristiques des analyses 56 et 89 ont changé tout comme celles de la 17, dans ce cas il suffit de changer la description N° 8 et les trois analyses deviendront correctes.

2. Les caractéristiques des analyses 56 et 89 n'ont pas changé, dans ce cas il faut créer une nouvelle description pour cette analyse 17 ou bien utiliser une description exitante et qui convienne à 17.

Dans les deux cas, lorsqu'on manipule une description, il est très utile de connaître la liste des analyses visées par cette description pour savoir sur quelles analyses va se répercuter la modification effectuée sur cette description. Cette liste des analyses visées est contenue dans le champ "LISTE".

---

28. La ième description contenue dans FILDSC a sa partie commentaire en position i dans FILCOM

5.4 Traitements possibles.

Le programme de gestion des analyses permet les opérations suivantes qui sont en fait les opérations classiques de traitement de fichier:

- Ajouter une analyse.
- Détruire une analyse.
- Consulter une analyse.
- Modifier une analyse.
- Ajouter une description d'analyse.
- Détruire une description d'analyse.
- Consulter une description d'analyse.
- Modifier une description d'analyse.
- Etablir une liste générale des descriptions.
- Etablir une liste détaillée par description.
- Etablir une liste générale des analyses.

Chapter 6

Dispatching batch.

6.1 Introduction.

Je vais expliquer ici, le fonctionnement général d'un dispatching batch automatisé mais il reste auparavant quelques notions supplémentaires à définir.

6.2 Notions supplémentaires.

6.2.1 Disposition de la table de travail de l'HAMILTON.

Comme cela est indiqué à l'appendice K, la table de travail de l'HAMILTON est découpée en 3 zones:

1. Zone des prélèvements à dispatcher appelée également rack primaire [29]. Ce rack comporte 500 positions numérotées disposées en quinconce d'où son nom également de ce rack en quinconce.

2. Zone des cupules réceptrices [30]; cette zone porte également le nom de rack secondaire. Les positions de ce rack sont disposées rectangulairement en 20 lignes et 12 colonnes.

3. Zone des tampons

---

29. Ce terme désigne un support en métal ou en plastique percé de trous pour y disposer des cupules ou des éprouvettes

30. autre nom des godets en plastique

6.2.2 Fichiers

Les deux fichiers décrits ci-après ne disposent pas de programme spécifique assurant leur gestion car leur structure est très simple et ils ne sont que très rarement modifiés. Ils sont toutefois accessibles et modifiables via un éditeur de texte classique, ce qui est amplement suffisant.

6.2.2.1 Fichier TAMPON.

Ce fichier contient les noms des différents tampons pouvant être utilisés lors du dispatching ainsi que les coordonnées de leur position matérielle sur la table de travail de l'HAMILTON. Le fichier se compose de n (nombre de tampons) triplets de lignes du type:

nom_du_tampon
coordonnée en x
coordonnée en y

6.2.2.2 Fichier POSREF.

Ce fichier permet de définir à quelle position du rack en quinconce doivent se placer les prélèvements, lors du dispatching, suivant la lettre de leur référence. Le fichier contient les 26 lettres de l'alphabet avec la position de base pour chaque référence commencant par cette lettre. C'est-à-dire:

A
0
B
0
.
.
J
1
.
.
K
0
.
.
N
476
.
.

Z
351

Ceci signifie

- que les lettres A, B et K ne sont pas utilisées,
- que les références commençant par J se mettront aux positions 1,2,3,...
- que les références commençant par N se mettront aux positions 476,477,...

Il existe un programme qui donne, pour un fichier FILREF donné un listing spécifiant pour chaque référence de ce fichier sa position exacte dans le rack en quinconce.

6.2.3 Groupe d'analyses.

La notion de groupe d'analyses doit être bien définie pour comprendre le fonctionnement et l'utilité du programme de dispatching. Lorsqu'il sera question de "groupe d'analyses" dans ce qui suit, if faut entendre pas là

> une série d'analyses dont, lors du dispatching, les quantités nécessaires respectives de prélèvement et/ou de tampon se retrouveront cumulées dans la même cupule du rack secondaire.

L'utilité d'une telle notion est la suivante: prenons le cas d'un technicien ou d'une machine qui effectue toute une série d'analyses différentes (ex:50). Si pour un patient on doit effectuer 30 de ces analyses, il est de loin préférable que le technicien ou la machine reçoive une seule cupule avec la somme des volumes nécessaires pour effectuer ces 30 analyses plutôt que de recevoir 30 cupules différentes avec chaque fois le volume nécessaire pour une seule analyse.

Les raisons en sont les suivantes:

- Si on compte 50 patients de ce type par jour, cela fait dans le premier cas 1500 cupules contre 50 dans le deuxième cas ce qui représente une fameuse économie de manipulations.
- Cela permet de réduire les volumes de sérum nécessaires: lorsqu'on dit que pour telle analyse il faut 100 microlitres de sérum, en fait l'analyse elle-même n'en utilise peut-être que 50 microlitres mais les 50 autres sont nécessaires car il y a des pertes (une partie du sérum adhère aux parois de

la cupule, une partie reste au fond car on ne sait pas tout attraper [31], ...

Si on groupe les 30 analyses dans une cupule, les pertes ne se font qu'une seule fois.

Ce regroupement dans une même cupule appelle certaines considérations:

- Il est absolument évident que tous les volumes de sérum cumulés dans une même cupule doivent provenir du même patient.

- Il n'est pas possible de regrouper des analyses pour lesquelles les tampons et/ou les dilutions sont différentes. Exemple: si on regroupe les analyses

  * X (300 microl. de sérum + 500 microl. d'eau physiologique (tampon))

  * et Y (300 microl. de sérum + 500 microl. de PBS (autre tampon))

  dans la même cupule, cela risque de provoquer certains mélanges indésirables tout comme si on regroupe les analyses

  * W (30 microl. de sérum + 100 microl. du tampon xyz)

  * et Y (30 microl. de sérum + 500 microl. du même tampon xyz)

  dans la même cupule, on risque d'avoir des dilutions fantaisistes.

Il faut enfin signaler qu'un groupe d'analyses peut n'en contenir qu'une seule.

---

31. Ceci est surtout vrai pour les machines qui vont aspirer le liquide dans la cupule. Le technicien lui, à la rigueur, peut transvaser le liquide en le versant. ce qui lui permet de ne rien laisser au fond.

6.3 Principe général.

Le programme doit permettre d'effectuer du dispatching batch c'est-à-dire: à partir de diverses informations aller pomper dans le rack primaire[32] certaines quantités, éventuellement pomper un certain volume d'un tampon particulier et recracher le tout à un endroit déterminé du rack secondaire.

6.4 Specifications.

6.4.1 Entrées.

Le programme de dispatching utilise 6 des fichiers décrits plus haut à savoir:

1. FILREF: Fichier des références. (Cfr 4.2.3)
2. FILDDE: Fichier des demandes. (Cfr 4.2.3)
3. POSREF: Fichier des positions de base. (Cfr 6.2.2.2)
4. TAMPON: Fichier signalétique des tampons. (Cfr 6.2.2.1)
5. FILANA: Fichier des analyses. (Cfr 5.3)
6. FILDSC: Fichier des descriptions d'analyse. (Cfr 5.3)

Outre ces fichiers, le programme travaille essentiellement sur les groupes d'analyses[33] qui lui sont fournis par l'utilisateur: Pour chaque groupe d'analyses à dispatcher, il est nécessaire de spécifier:

---

32. contenant tous les prélèvements de la journée

33. Le programme permet de dispatcher à la fois un maximum de 5 groupes d'au plus 40 analyses chacun.

1. L'option choisie. 3 options sont possibles:
    1. dispatcher prélèvement+tampon(Dilution).
    2. dispatcher uniquement le tampon.
    3. dispatcher uniquement le prélèvement.
2. La colonne du rack secondaire à partir de laquelle seront dispatchées les analyses de ce groupe.
3. Le nombre de colonnes de ce même rack que l'on peut utiliser pour ce groupe.
4. La liste des analyses à dispatcher dans ces colonnes.

6.4.2 Sorties.

Le programme fournit un tableau permettant de retrouver en quel endroit il a dispatché telle référence. Ce tableau est du type:

- nro_colonne * nro_ligne * référence *

6.5 Exemple.

Un exemple permettra de mieux comprendre le fonctionnement de ce programme.

Les prélèvements de la journée sont disposées dans le rack en quinconce, le rack de droite est rempli de cupules vides et les tampons sont disposés au bon endroit.

Soit le groupe 1 introduit par l'utilisateur:

- Données introduites:
    * option:Dilution
    * position: 5
    * nombre de colonnes: 2
    * analyses: 4 25 158 266
- Résultat: le programme va prendre la première référence, c'est-à-dire le premier prélèvement, regarder la ou lesquelles des 4 analyses (4 25 158 266) font l'objet d'une

demande pour cette référence et sommer les quantités respectives nécessaires pour ces analyses. Puisque l'option est "dilution", il va déposer dans la première cupule de la colonne 5, la quantité totale calculée de tampon [34] ainsi que la quantité totale calculée de prélèvement.

- On va donc retrouver dans cette cupule la quantité de produit nécessaire pour effectuer celles des 4 analyses qui ont été demandées pour la première référence.

- Le programme va prendre ensuite la deuxième référence, effectuer les mêmes cumuls, et déposer les produits dans la deuxième cupule de la colonne 5. Le traitement va se poursuivre pour toutes les autres références de la journée en utilisant les cupules des colonnes 5 et 6 puisque le nombre de colonnes utilisées est de 2.

Une description plus systématique du programme sera donnée au chapitre 11.

---

34. Le programme aura eu soin de vérifier que les tampons et les dilutions pour ces 4 analyses sont identiques.

Chapter 7

Le problème des COBAS.

7.1 Introduction.

Le problème a déjà évoqué en 2.4.1.

Rappelons qu'il s'agit de deux machines d'analyse qui effectuent plus de 500 analyses par jours. Dans la situation de départ, les résultats sont imprimés par les machines et recopiés par les techniciens sur des cartes-réponses qui sont ensuite perforées par les encodeuses. (Cfr 2.3.1)

7.2 Proposition d'automatisation.

7.2.1 Au niveau matériel.

On se propose d'utiliser un Xérox relié aux deux machines pour en capter les résultats par ligne directe puisque chaque COBAS dispose d'une porte de sortie. Cette connexion se fait à travers un contrôleur multiportes: cet appareil dispose d'un coté d'une porte maîtresse qui sera reliée au Xérox et de l'autre coté de 4 portes esclaves dont deux seront reliées aux COBAS.

Le principe de fonctionnement de cet appareil est tout simple; il s'agit d'un genre d'aiguillage. La porte maîtresse peut décider de s'aiguiller sur telle ou telle porte esclave et établir ainsi une liaison entre l'appareil relié à cette porte esclave et celui relié à la porte maîtresse. De plus chacune des 4 portes esclaves possède une mémoire de 256 caractères. Ceci est très utile dans le cas suivant: supposons que la porte maîtresse est aiguillée sur la porte esclave 2; pendant ce temps, si l'appareil relié à la porte esclave 4 veut envoyer 200 caractères vers l'appareil relié la porte maîtresse, il peut le faire sans attendre d'y être aiguillé: les 200 caractères seront mémorisés dans la mémoire de la porte 4 et quand la porte

maîtresse s'aiguillera vers cette porte 4 elle pourra récupérer les 200 caractères.

7.2.2 Au niveau programme.

Le programme réalisé est assez volumineux car il doit effectuer les traitements suivants:

1. A partir des fichiers de demandes (FILREF et FILDDE) il doit créer des tableaux de travail pour les techniciens qui travaillent sur les COBAS.

   Avant que le programme pour les COBAS ne soit opérationnel, ces tableaux étaient créés par le programme de création de tableaux mais comme le programme COBAS effectue toute une série d'opérations sur ces tableaux, il est préférable qu'il les aie en mémoire et c'est pourquoi il les constitue lui-même à partir des fichiers de demandes issus de l'encodage. Par la même occasion ces tableaux ont été améliorés afin d'optimiser et de faciliter le travail des techniciens.

2. Il doit réceptionner les résultats envoyés par les deux COBAS, faire certaines vérifications sur les résultats des tests de contrôle [35], et enfin archiver les résultats reçus.

3. Il doit générer les cartes-résultats

4. Il doit enfin assurer une série de fontions annexes supplémentaires pour faciliter certaines opérations de consultation ou de correction de résultats.

Ce programme n'entre pas tout à fait dans la cadre de ce mémoire qui, rappelons le, visait à automatiser le dispatching et c'est pourquoi je ne m'attarderai pas davantage dessus.

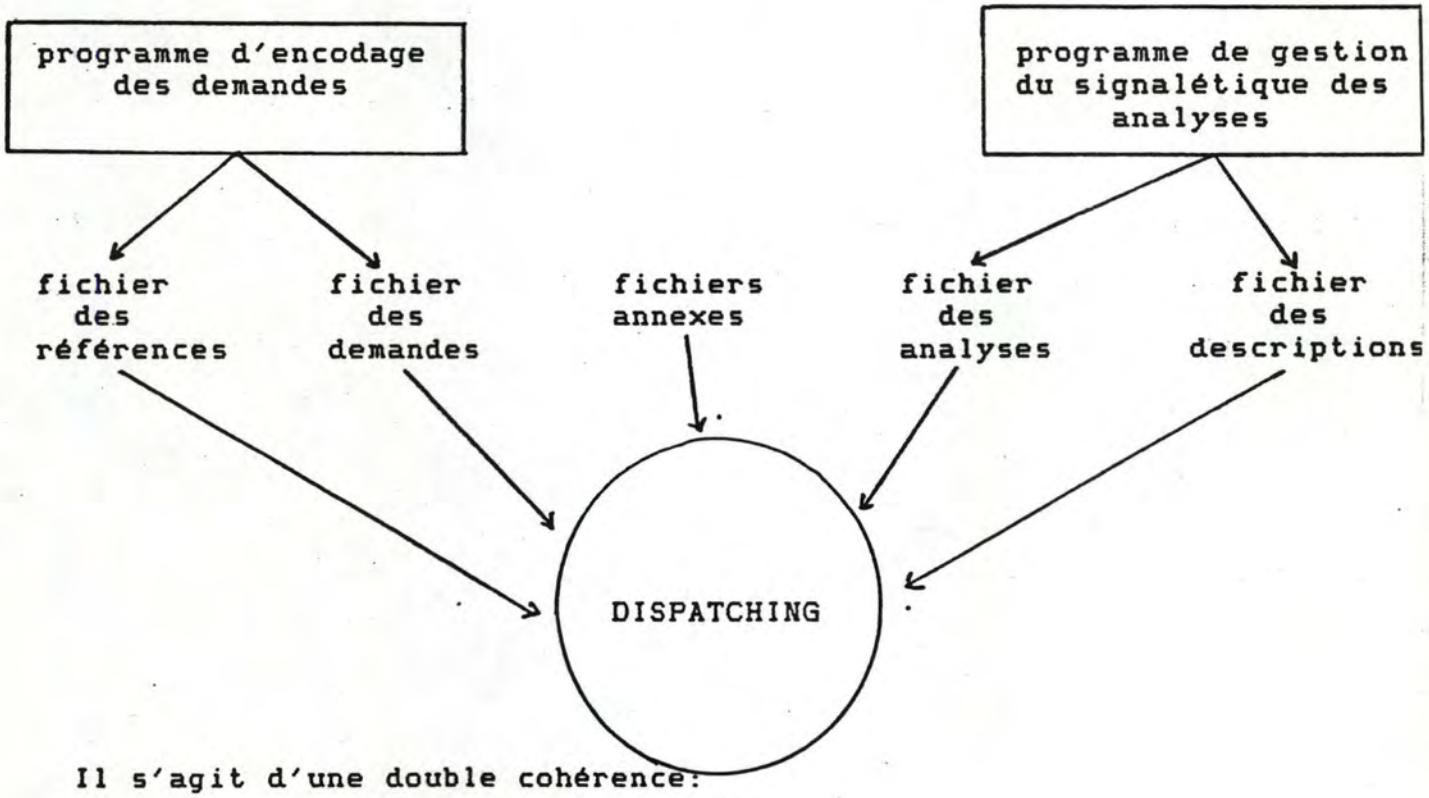
---

35. Chaque fois (ou presque) qu'on fait une analyse sur une série de prélèvements, on ajoute 1 ou 2 faux prélèvements qui sont en fait des liquides de contrôle pour lesquels les résultats théoriques d'analyse sont connus. Il suffit dès lors de vérifier si la machine donne pour ces contrôles un résultat dont la valeur se situe dans une fourchette centrée autour de la valeur théorique. Si ce n'est pas le cas, cela veut dire que les réactifs sont trop vieux ou qu'il y a un problème à la machine.

Chapter 8

Conclusion.

L'ensemble des programmes écrits forme un tout cohérent comme on peut le voir dans le schéma ci-dessous:



Il s'agit d'une double cohérence:

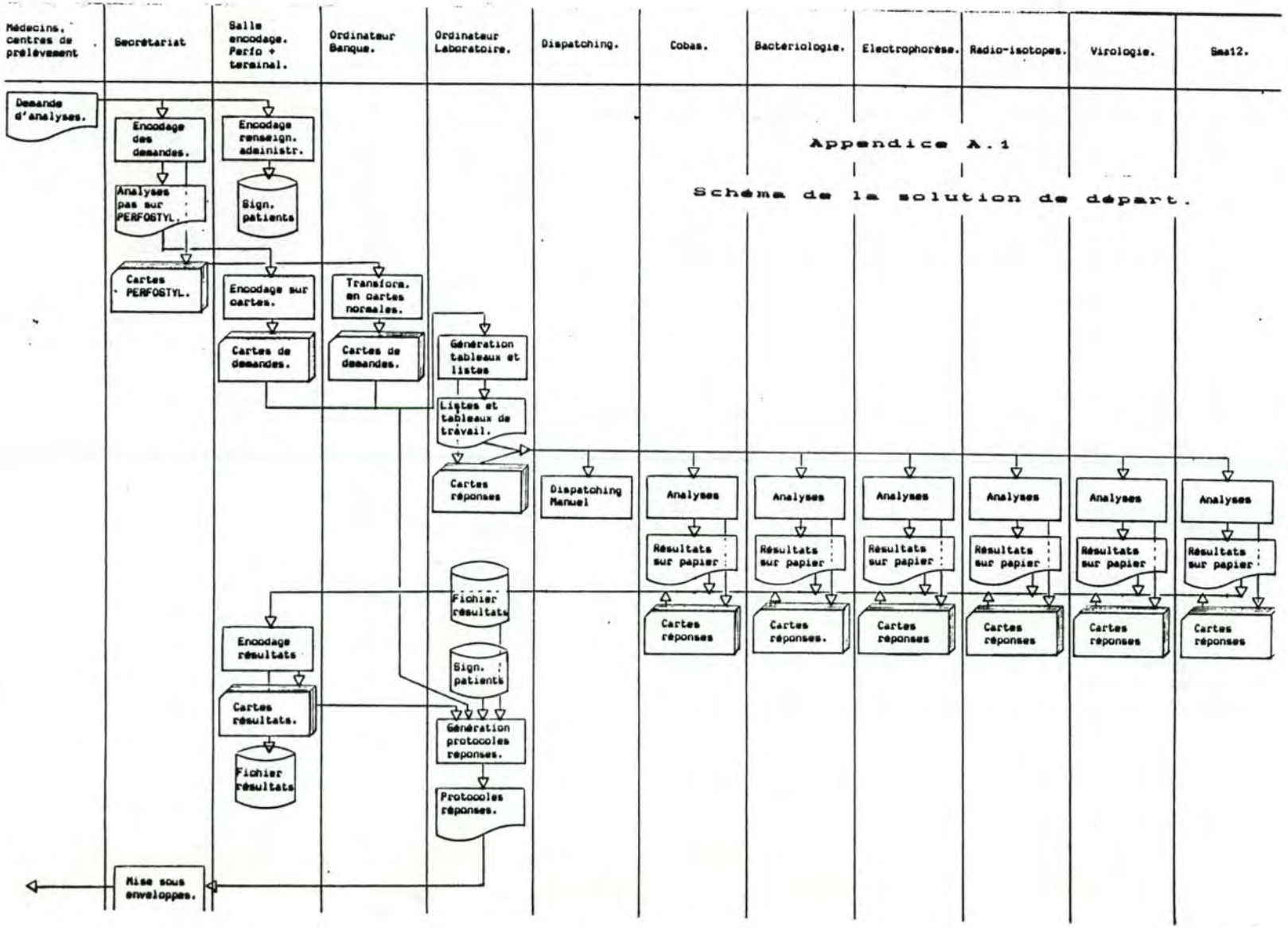
- horizontale car l'ensemble des programmes écrits forme un bloc autonome.
- verticale car ce bloc s'intègre parfaitement dans le processus d'analyse (Cfr note 7 page 13) du laboratoire.

Ce mémoire a été très intéressant dans la mesure où il a touché à de nombreux domaines: robotique, micro-informatique, communications par ligne, langage C ... .

Son intérêt s'est encore accru par le fait que tous les programmes sont actuellement pleinement opérationnels. Seul, le dispatching n'est pas encore utilisé, faute de machine. Le caractère "user-friendly" de tous ces programmes et leur souplesse expliquent sans doute l'utilisation intense qui en est faite.

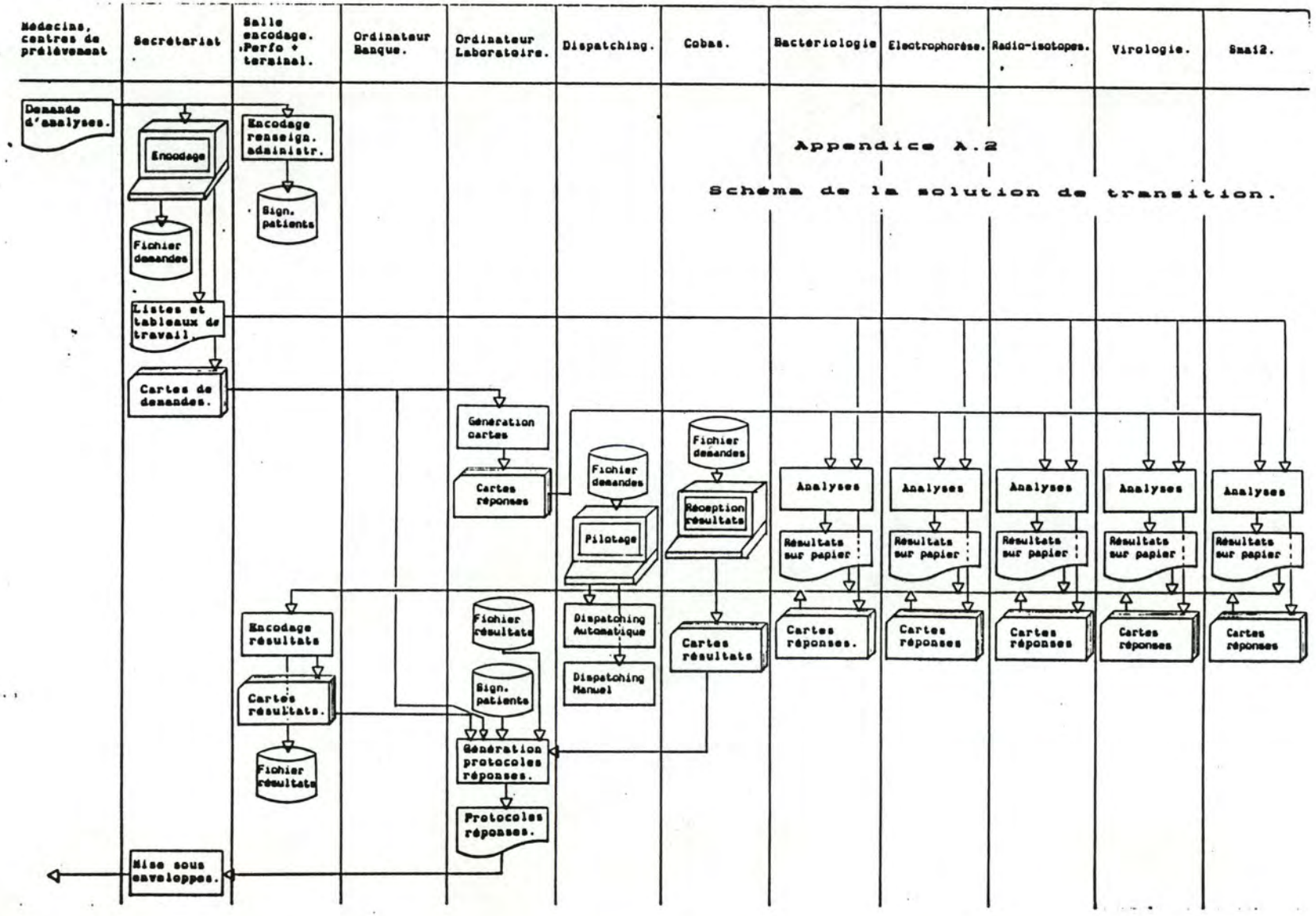
Appendix A

Schémas des flux.



Appendice A.1

Schéma de la solution de départ.

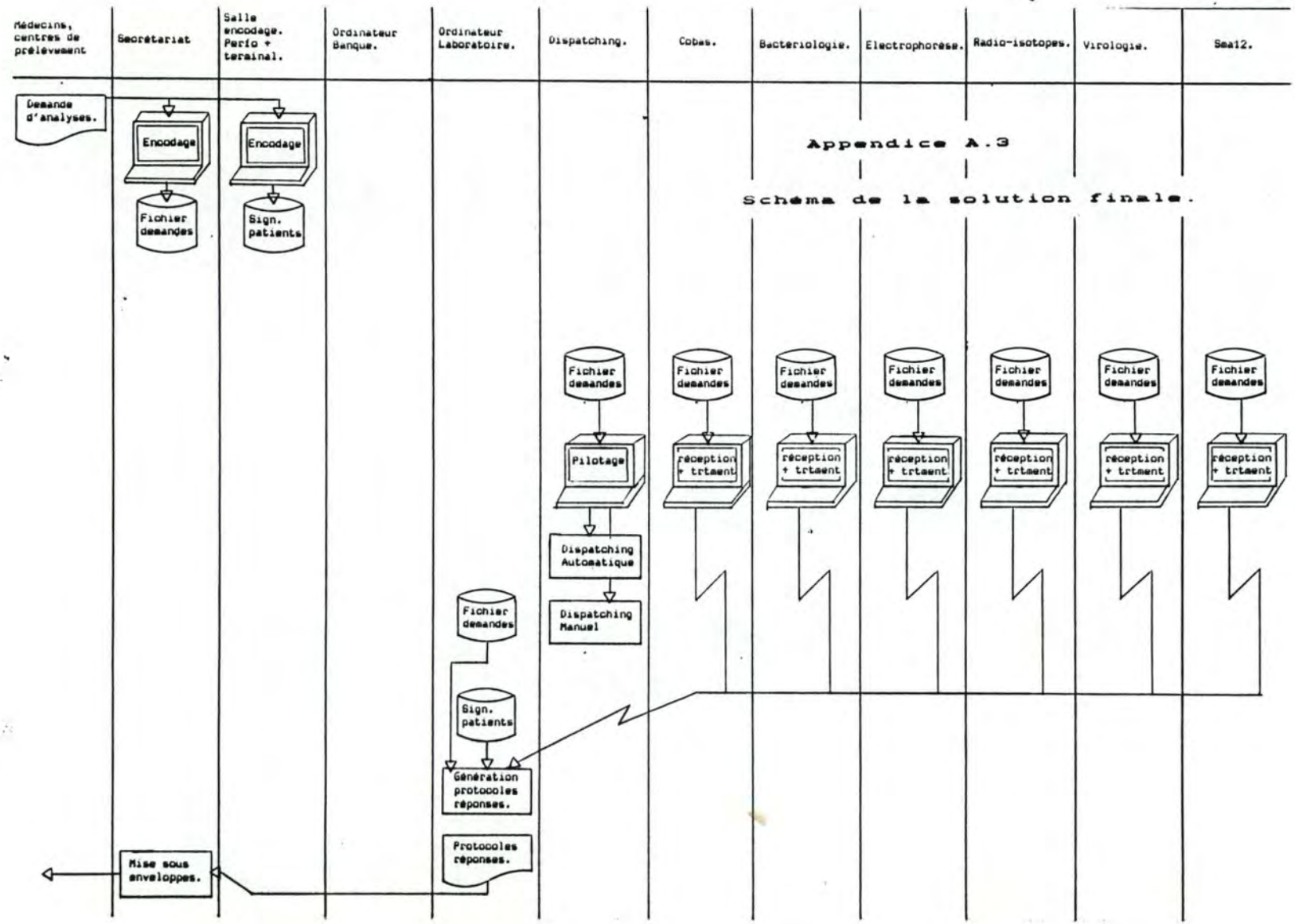


Appendice A.2

Schéma de la solution de transition.

Appendice A.3

Schéma de la solution finale.



Appendix B

Feuille de demandes.

- 56 -

BACTERIOLOGIE

URINES USUELLES

144 0 Protéines. Albumine.
145 0 Sucre.
146 0 Ex. microscopique.
165 0 Gram.
162 0 Culture.
164 0 Antibiogramme.
166 0 Cult. mycose + Identific.

147 0 Densité.
148 0 Acétone.
150 0 Sels biliaires.
152 0 p.H.
158 0 Electrophorèse.
151 0 Pigments biliaires.
167 0 Test de grossesse.
570 0 Cellules néo.

URINES 24 H.

159 0 Sodium.
160 0 Potassium.
154 0 Calcium.
161 0 Chlore.
155 0 Phosphore
156 0 Acide urique.
326 0 Urée.
149 0 Créatinine.
157 0 Amylase.
432 0 Cortisol.
138 0 Aldostérone.
0 17-cétos totaux.
0 17-cétos chromato.
0 17 OH.
203 0 Ziehl homo.
163 0 Culture B.K.

URINES 24 H. ACIDIFIEES

0 Catécholamines.
0 VMA.
0 5 HIAA (Serotonine).
0 Adrénaline.
0 Noradrénaline.

URINES 24 H. ALCALINISEES

359 0 β2 Microglobuline.

CLAIRANCE - CLEARANCE ••••

084 0 Urée.
086 0 Créatinine.
087 0 Acide urique.
0 Amylase.

FROTTIS DE GORGE

198 0 Gram.
200 0 Culture + Méningocoque.
202 0 Antibiogramme.
166 0 Cult. Mycose + Identific.

EXPECTORATIONS

198 0 Gram.
200 0 Culture
202 0 Antibiogramme.
166 0 Cult. mycose + Identific.
203 0 Ziehl homo.
201 0 Culture B.K.
571 0 Cellules néo.

SELLES

191 0 Examen direct.
188 0 Culture.
190 0 Antibiogramme.
440 0 Campylobacter.
189 0 Sang.
186 0 Parasites.
187 0 Résidus.
294 0 Antigène rotavirus.

PUS, SONDES, DRAINS

198 0 Gram.
204 0 Culture aérobie.
533 0 Culture anaérobie
+ Identific. anaérobies.
202 0 Antibiogramme.

PUS URETHRAL ou VAGINAL

531 0 Ex. microscopique à frais.
198 0 Gram.
204 0 Cult. aérobie + Gonocoque
533 0 Culture anaérobie
+ Identific. anaérobies.
202 0 Antibiogramme.
166 0 Cult. mycose + Identific.

SPERME

198 0 Gram.
461 0 Numération + Mobilité.
462 0 Formule.
204 0 Cult. aérobie + Gonocoque
202 0 Antibiogramme.
0 Cellules néo.

LIQUIDE PONCTION

002 0 Globules rouges.
003 0 Globules blancs.
004 0 Formule.
042 0 Acide urique.
0 Cristaux.
018 0 Protéines.
331 0 Rivalta.
198 0 Gram.
204 0 Culture aérobie.
533 0 Culture anaérobie.
+ Identific. anaérobies.
202 0 Antibiogramme.
572 0 Cellules néo.

COL UTERIN

0 Cellule néo.

CLINIQUE MEDICALE :

Analyses supplémentaires :

LOT 5 - 3-84 - 30.000

S.V.P. COLLER UNE VIGNETTE. MERCI D'AVANCE.

Organisme Assureur : ____
N° d'inscription : ____
TITULAIRE. Nom. : ____
: ____

PATIENT. Nom. : ____
: ____
N° d'inscription : ____
CT 1 : ____
CT 2 : ____

☐☐☐☐ ☐☐☐☐ ☐☐☐☐ ☐☐☐☐

Titulaire - Conjoint - Enfant - Ascendant.

Date de naissance : ____
Sexe : Masculin - Féminin.
Veuf(ve) - Invalide - Pensionné(e) - Orphelin(e).
Adresse : ____
: ____
: ____
: ____
Date du prélèvement : ____

Nombre d'analyse demandées : ____
Date de la demande : ____
Signature et cachet du médecin.

N/R. : ____

HEMATOLOGIE

000 0 V.S. —
003 0 Globules blancs. *
004 0 Formule leucocytaire. *
074 0 Plaquettes. *
001 0 Hémoglobine. *
005 0 Hématocrite. *
002 0 Globules rouges. *

060 0 Réticulocytes. *
061 0 Ponctuations basophiles * Eosinophiles.
340 0 Vitamine B 12.
341 0 Ac. Folique (AF).
343 0 AF Erythrocytaire. *
035 0 Fer sérique. 0 Cap. fix.
338 0 Ferritine.
545 0 Transferrine.
544 0 Haptoglobine.
490 0 Kleihauer. *

IMMUNO-HEMATOLOGIE

054 0 Groupe sanguin ABO.
055 0 Facteur Rh. (D).
056 0 Coombs direct.
057 0 Coombs indirect.

HEMOSTASE * *

072 0 Thrombotest. (Adulte).
315 0 Thrombotest. (Enfant).
069 0 Prothrombine (Adulte).
314 0 Prothrombine (Enfant).
071 0 T. de Céphaline.
075 0 Fibrinogène.
387 0 Antithrombine III Activité.
381 0 Facteur VIII Activité.
389 0 Aggrégabilité plaquet. (RV)
066 0 Temps de saignement (RV)
067 0 Temps de coagulation.
0 Temps de Howell.

ENZYMES

026 0 S.G.O.T.
027 0 S.G.P.T.
041 0 C.P.K.
246 0 Iso - C.P.K.
033 0 L.D.H.
017 0 Iso - L.D.H.
034 0 Amylase.
040 0 Lipase.
039 0 Aldolase.
052 0 L.A.P.
049 0 O.C.T.
063 0 G.G.T.
321 0 5'nucléotidase.
215 0 Cholinestérase.
028 0 Phosphatase alcaline (PA)
248 0 ISO - PA.
029 0 Phosphatase acide totale
050 0 Phosphatase prostatique.

TESTS HEPATIQUES

030 0 Bilirubine totale.
031 0 Bilirubine directe.
020 0 Thymol.
024 0 Kunkel zinc.
259 0 Ammonium *

GLUCIDES

007 0 Glycémie. (G). — —
007 0 G. Post-prandiale (H:)
316 0 Hb Glycosylée. *
421 0 Insulinémie (I).
334 0 Peptide-C (PC).
989 0 Hyperglycémie (RV).
422 0 I durant Hyper (RV).
334 0 PC durant Hyper (RV).
215 0 C2H5-OH. — —

REIN

006 0 Urée.
036 0 Créatinine.
042 0 Acide urique.
349 0 Rénine angiotensine. *

IONOGRAMME

096 0 Sodium.
097 0 Potassium.
098 0 Chlore.
099 0 Réserve alcaline.
102 0 Calcium.
103 0 Phosphore.
105 0 Magnésium sérique (Mg).
327 0 Mg Erythrocytaire. * * *
047 0 Cuivre.

INFLAMMATION

000 0 VS. —
045 0 C.R.P.
043 0 R.A. test Latex.
044 0 Waaler Rose.

542 0 α 2-macroglobuline.
541 0 Céruloplasmine.
550 0 Complément C'3.
551 0 Complément C'4.
544 0 Haptoglobine.
062 0 Orosomucoïde.
075 0 Fibrinogène. * *

PROTEINES

018 0 Protéines totales.
019 0 Electrophorèse.
536 0 IgA.
535 0 IgG.
537 0 IgM.

LIPIDES

011 0 Lipidogramme.
008 0 Cholestérol.
023 0 Triglycérides.
010 0 Lipides.
037 0 Phospholipides.
038 0 Ac. gras libres.
016 0 HDL cholestérol et cholestérol lipoprotéine.

SEROLOGIE

113 0 B.W.
328 0 V.D.R.L.
088 0 T.P.H.A.
094 0 Tréponème fluorescence.
305 0 Chlamydia.
307 0 Gonocoque.
279 0 Herpès 1.
280 0 Herpès 2.

124 0 Toxo. Agglutination dir.
119 0 Toxo. Hémagglutination.
125 0 Toxo. Fluorescence.
250 0 Toxo. Complément.
117 0 Toxo. IgM.

110 0 Paul-Bunnell qualit.
111 0 Paul-Bunnell quantit.
287 0 (IgG + IgM) anti EBV.
266 0 (IgM) anti-capside EBV.

118 0 Rubéole HAI.
274 0 Rubéole FC.

046 0 A.S.L.
245 0 Exoenzymes du strepto.
241 0 Antistreptokinase.
255 0 Anti-protéine M.
258 0 Antihyaluronidase.
0 Antistaphylolysine.

306 0 Chlamydia psittaci.
308 0 Yersinia.
108 0 Widal.
109 0 Wright.
192 0 Listeria.

GENETIQUE

543 0 α 1 Antitrypsine.
390 0 Phénotype α 1 Antitrypsine.
0 HLA (RV). * * *
0 Caryotype (RV). * * *

AUTO ANTICORPS

056 0 Anti-GR (Coombs). *
370 0 Anti-noyau (FAN).
374 0 Anti-DNA.
371 0 Anti-mitochondries.
372 0 Anti-muscules lisses (ML).
373 0 Anti-reticuline.
374 0 Anti-cellules pariétales (P).
375 0 Anti estomac (ML + P).
376 0 Anti myocarde.
095 0 Le cellé. *
550 0 Complément C'3.
551 0 Complément C'4.

MEDICAMENTS

394 0 Digoxine.
395 0 Digitoxine.
106 0 Lithium.
618 0 Théophylline.
609 0 Phénobarbital.
612 0 Diphenylhydantoïne.
617 0 Acide valproïque.

ONCOLOGIE SERIQUE

050 0 Phosphatase prostatique.
452 0 α Fœtoprotéine.
435 0 C.E.A.
358 0 β 2 Microglobuline.
252 0 Lysozyme.

IMMUNOCHIMIE SPECIALE

0 Viscosité
0 Immunoélectrophorèse.
112 0 Agglutinines froides.
243 0 Cryoglobuline.

ALLERGIE DEPISTAGE

0 IgE totales.
0 Graminées.
0 Herbacées 1.
0 Herbacées 2.
0 Arbres 1.
0 Arbres 2.
0 Animaux.
0 Moisissures.
0 Dermatophagoïdes.

ALLERGIE SPECIFIQUE

Voir feuille annexe.

* Tube EDTA ou bouchon mauve.
** Tube citrate ou bouchon bleu.
*** Tube héparine-lithium ou bouchon vert.
**** Urines 24 h + 1 tube sérum.
— Tube citrate ou bouchon orange.
—— Tube fluorure ou bouchon gris.
RV : Sur rendez-vous au laboratoire.
ZF : Zone Fasciculée.
ZR : Zone Réticulée.

VIRUS RESPIRATOIRES

265 0 Adénovirus. (R1)
268 0 Mycoplasma. (R2)
269 0 Enterovirus. (R3)
282 0 Influenza A. (R4)
283 0 Influenza B. (R5)
285 0 Parainfluenza 1. (R6)
286 0 Parainfluenza 3. (R7)
292 0 V. resp. syncytial. (R8)

VIRUS NEUROLOGIQUES

271 0 Cytomégalovirus. (N1)
289 0 Poliovirus. (N2)
296 0 Picorna virus. (N3)
281 0 Zona-varicelle. (N4)
279 0 Herpes 1. (N5)
288 0 Oreillons. (N6)
293 0 Rougeole. (N7)
299 0 Coxsackie B 1. (N8)
290 0 Coxsackie B 2 - B 5. (N9)
216 0 Enterovirus. (N10)
287 0 IgG + IgM anti EBV. (N11)
266 0 IgM anti-cap. EBV. (N12)

VIRUS (ERUPTION, GANGLION)

217 0 Cytomégalovirus. (E1)
218 0 Mycoplasma. (E2)
219 0 Chlamydia. (E3)
220 0 Zona varicelle. (E4)
222 0 Herpes 1. (E5)
223 0 Rougeole. (E6)
225 0 Coxsackie B 1. (E7)
233 0 Coxsackie B 2 - B 5 (E8)
251 0 Rubéole. (E9)
236 0 Rickettsia burnetti. (E10)

VIRUS (PLEURO, PERICARDE)

275 0 Coxsackie B 1. (P1)
300 0 Coxsackie B 2. (P2)
301 0 Coxsackie B 3. (P3)
302 0 Coxsackie B 4. (P4)
303 0 Coxsackie B 5. (P5)

VIRUS DE L'HEPATITE

442 0 (IgM + IgG) Hépatite A.
444 0 (IgM) Hépatite A.
123 0 Antigène S Hépatite B.
441 0 Anticorps S Hépatite B.
443 0 Anticorps c Hépatite B.
445 0 Antigène e Hépatite B.
446 0 Anticorps e Hépatite B.

GROSSESSE

419 0 H.C.G. qualitative.
420 0 β-H.C.G. dosage.
414 0 Progestérone.
425 0 17 OH Progestérone.
412 0 Œstrone. (E1)
410 0 Œstradiol. (E2)
411 0 Œstriol total. (E3)
452 0 α Fœtoprotéine.
392 0 SP1 Glycoprotéine.
346 0 H.P.L.

HORMONES ♀+

417 0 F.S.H.
416 0 L.H.
418 0 Prolactine.
410 0 Œstradiol. (E2)
414 0 Progestérone.
434 0 Sex binding globulin.

HORMONES ♀

415 0 Testostérone totale.
438 0 Dihydrotestostérone.
428 0 DHEA Sulfate.
426 0 Δ4 Androstènedione.
434 0 Sex binding globulin.

HYPOPHYSE

379 0 Hormone croissance.
402 0 TSH.
0 TRH - TSH épreuve. (RV)
418 0 Prolactine.
0 TRH - Prolactine. (RV)
0 LH - RH épreuve (RV)

PARATHYROIDE

398 0 Parathormone.
0 Vitamine D.
102 0 Calcium.
103 0 Phosphore.

SURRENALE

347 0 ACTH. *
436 0 Aldostérone.
431 0 Cortisol-8 H. (ZF)
431 0 Cortisol-16 H.
433 0 Cortisol-épreuve (RV).
428 0 DHEA sulfate. (ZR)

THYROIDE

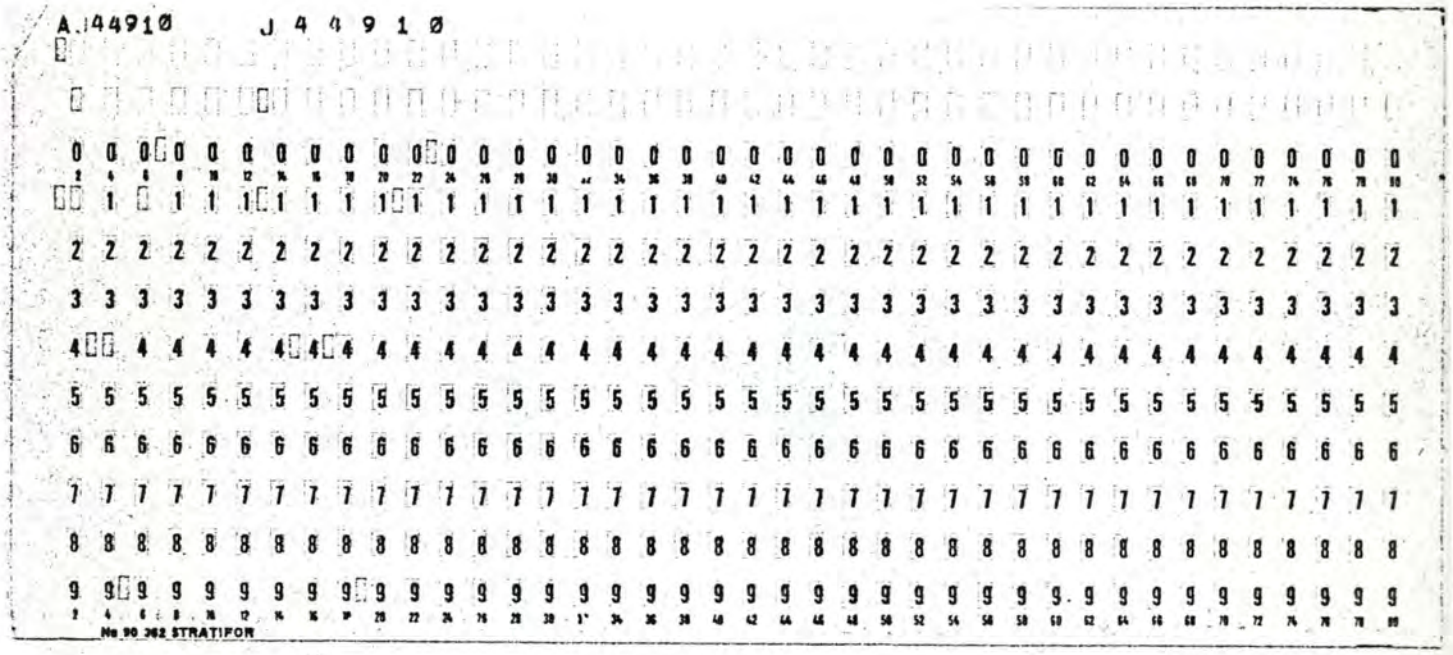
355 0 T4.
401 0 T3.
404 0 T3 R. Uptake.
402 0 T.S.H.
406 0 T.B.G.
403 0 Antithyrogl. (ACAT1).
405 0 Antimicrosome (ACAT2)
337 0 Thyroxine libre index.
356 0 Thyroxine liée index.
434 0 Sex binding globulin.

Appendix C

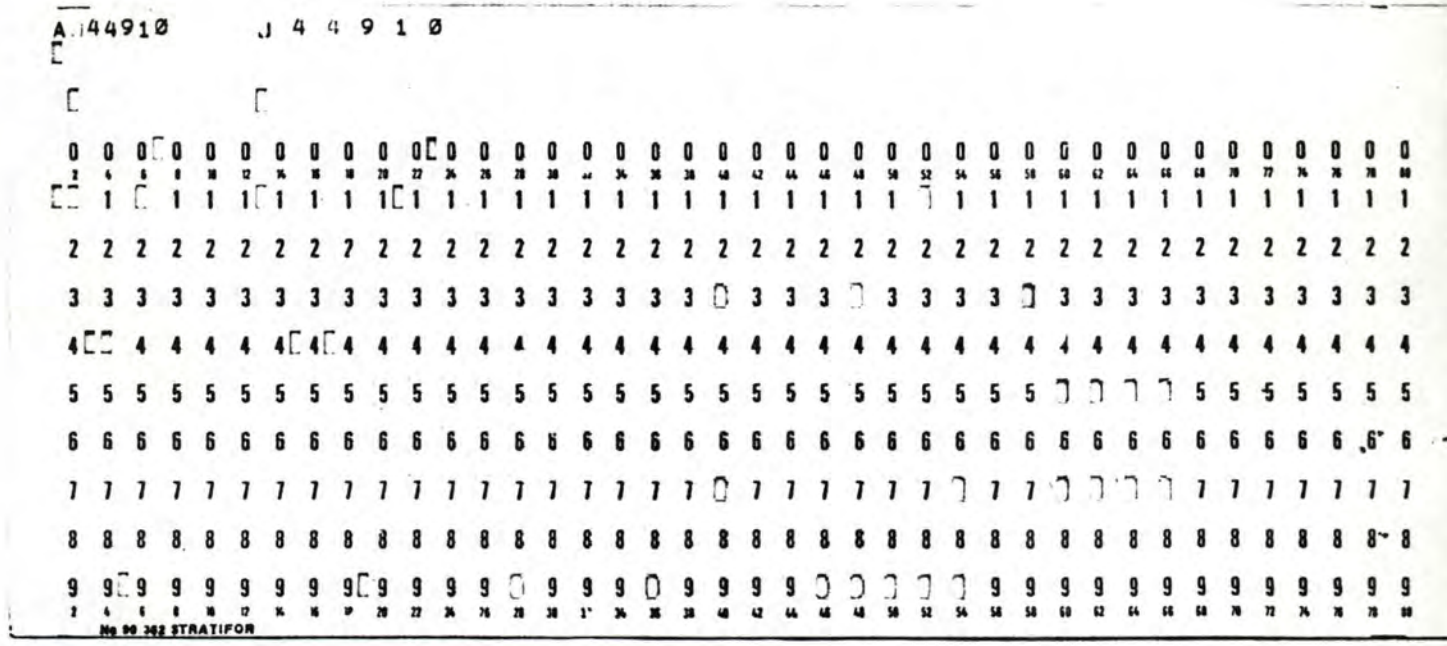
carte PERFOSTYL et grille.

VS	Hb.	Gl.R.	Gl.BL	F.L.	Hcte
Urée	glycémie	Cholest.	Cholest estér.	Lipides	Lipidogramme
				HDL	Iso LDH
Prot. tot	E°phorèse	Thymol	Takata	Hanger	Triglycérides
Kunkel zinc	phénol	Transaminases SGO	SGP	Phosphatases Alc.	Ac.
Bilirubine totale	directe	BSP	LDH	Amylase	Fer sérique
Créatinine	Phospholipides	Acide gras libre	Aldolase	Lipase	CPK
Acide urique	Ralest	Rose	C.R.P.	A.S.L	Cuivre
	OCT	Phosphatases prostatiques	H.B.D.H.	LAP.	Capacité en fer
Groupe	Rh.	Coombs direct	indirect	Compatibil	
Réticulo cytes	Ponctuation baso		γGT	LPX	Phases Alc. thermost
T.S.	T.C.	Lacet	P.T.T.	Héparine	Céphaline
Thrombotest	Howell	Plaquettes	Fibrinogène	Caillot	Cholinestérase
Fact. II	Fact IV	Fact VII	Cortisol plasm.	Catéchol. plasm.	Adrénaline plasm.
Clearance urée	P.S.P.	Clearance créatinine	Clearance ac. uriq.	B.W.	Hyperglyc.
Leuco. concent	L.E. test		Immuno E°phorèse	B.W.	L.E. Cells
Sodium	Potassium	Chlore	Réserve alcaline	P.H.	Ac. vanil. mandel.
Calcium	Phosphore	Plomb	Magnésium	Lithium	SHIAO
Widal	Wright	Paul Bunnel 1	Paul Bunnel 2	Agglutinines froides	B.W
Sérum	PAt	T3	TOXO	Rubéole	TOXO
Moëlle	Hémoculture		Antigène austral.	Toxo	Toxo
Ponction sternale	LCR Na L	LCR K	LCR Ca C	LCR P	LCR E°phorèse R
LCR Album.	LCR Sucre	LCR Éléments	LCR Chlore	LCR BW	Benjoin.
Aldostérone	Androstérone	17 OH	Corticoïdes totaux	Cortisol urinaire	
Alb. U	Sucre R	Microsc. I	Densité N	Acétone E	Créatinine S
Sels biliaires	Pigments biliaires	P.H.	Hémogl.	Ca.	P.
Acide urique	Amylase	E°phorèse	Na	K	Cl
Culture ordin.	Culture B.K.	A.T.B.	urines Gram	Mycoses	Grossesse
Phénol stéroïdes	Oestriol	Pregnandiol	Pregnantriol		
Addis	Calcul urinaire				
Tubage gastrique	Histamine				
Parasites S	Résidus E	Culture L	Sang L	A.T.B. E	Ex direct S
Listéria	Catécholamines	17 céto	Immuno diffusion	Chromato 17 céto.	
Gram	Ziehl	Culture ordin.	Culture BK	ATB	Ziehl homo
	Adrénaline urinaire	Métané-phrine			
Annulé	Attendre	BK en cours	Autres analyses	Autre médecin	A compléter

grille en plastic



non encore encodée



encodée

Appendix D

Feuille de programmation.

Ex 267-75

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38
J	4	7	0	5	1			9	0			7	5			2	1	9			8	6		2	8		5	4	5		5	4	6		5	4	7
J	4	7	0	5	2			5	3	3		5	3	4		5	3	5																			
J	4	7	0	5	3			2	4	5		2	4	6		3	5	1																			
N	2	1	7	2	7			3	7	5		3	7	7			2	8			2	9		5	7												
Z	7	2	9	2	1			2	4	1		2	4	2		2	4	3		9	7			2	8	1		2	8	2							

Appendix E

Petites listes de travail.

2/ 5/84 11:50:25

144 U.PROTEINES TOTALES

J59468
J59507
J59519
J59521
J59522
J59524
J59527
J59533
J59536
J59537
J59538
J59539
J59546
J59547
J59549
J59551
J59556
J59558
J59569
J59562

2/ 5/84 11:50:20

20 S.THYMOL

J59503
J59544
J59547
J59548
J59549

2/ 5/84 11:50:25

106 S.LITHIUM PLASMATIQUE

J59492

Appendix F

Tableaux de travail.

L I S T E 3

virologie 2/ 5/84 11:47:21

	306 ORN	307 GON	282 INa	283 INb	285 INc	PA1	PA2	286 PA3	288 ORE	293 223 ROU	292 RSV	ROT	274 RUB	308 Y3	Y9	289 POL
J59473	///	///		///		///	///	///	///	///	///		///	///	///	///
J59526	///	///	///	///	///	///	///	///	///	///	///	///	///	///	///	///
L10505	///		///	///	///	///	///	///	///	///	///	///	///	///	///	///

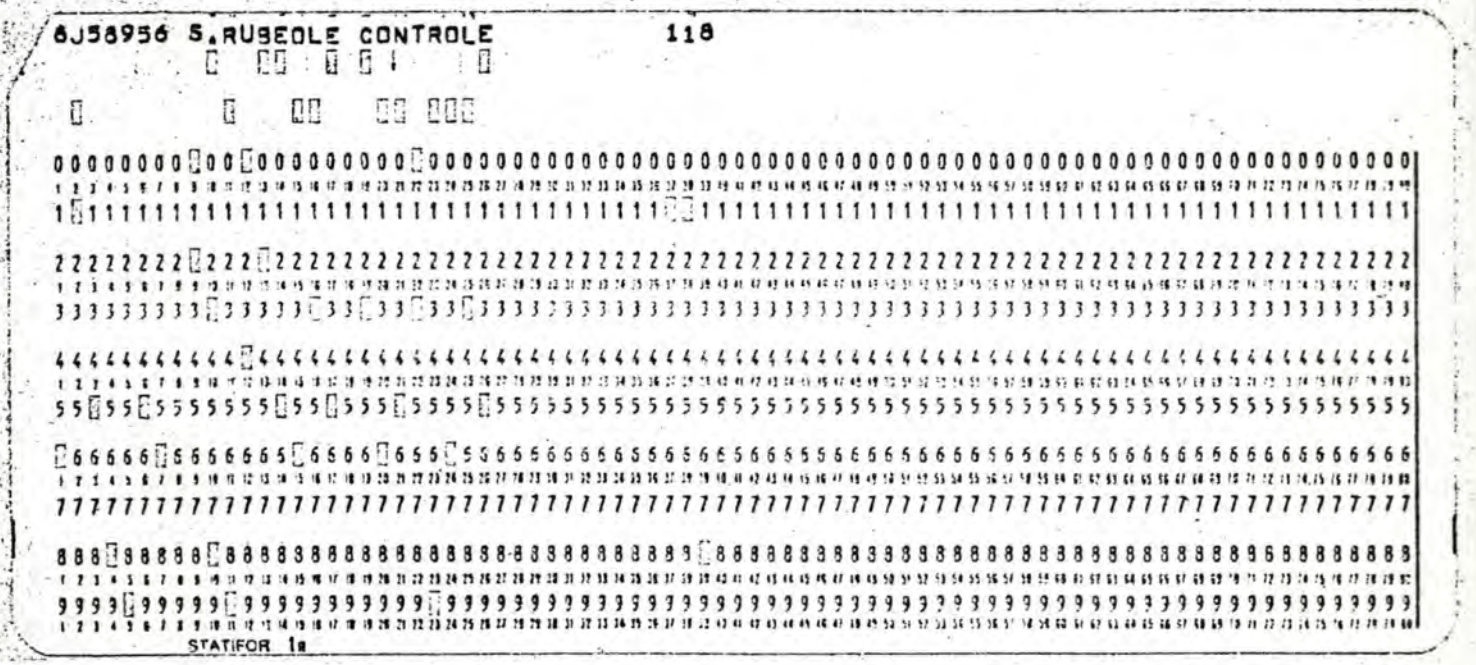
L I S T E 3

Nephelometre Technicon 2/ 5/84 11:47:27

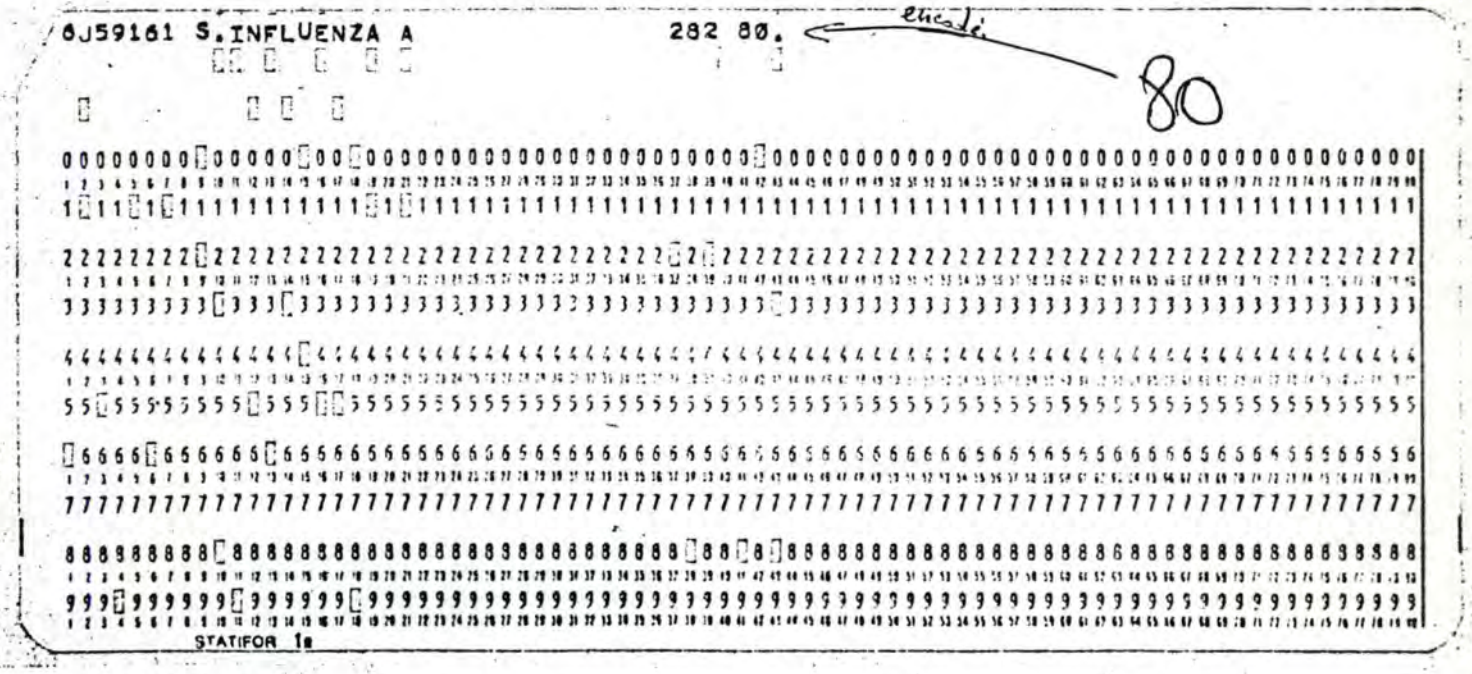
	53 CAP. FER	545 TRANSFER.	535 IgG	536 IgA	537 IgM
J59461		///	///	///	///
J59467	///		///	///	///
J59471		///	///	///	///
J59472		///	///	///	///
J59473	///	///	///		///
J59480			///	///	///
J59525		///	///	///	///
J59546		///	///	///	///
J59554	///	///			///
J59556		///	///	///	///
J59559		///	///	///	///
J59561		///	///	///	///

Appendix G

Carte-réponse et carte-résultat.



Carte réponse



Carte résultat

Appendix H

Schéma de l'HAMILTON.

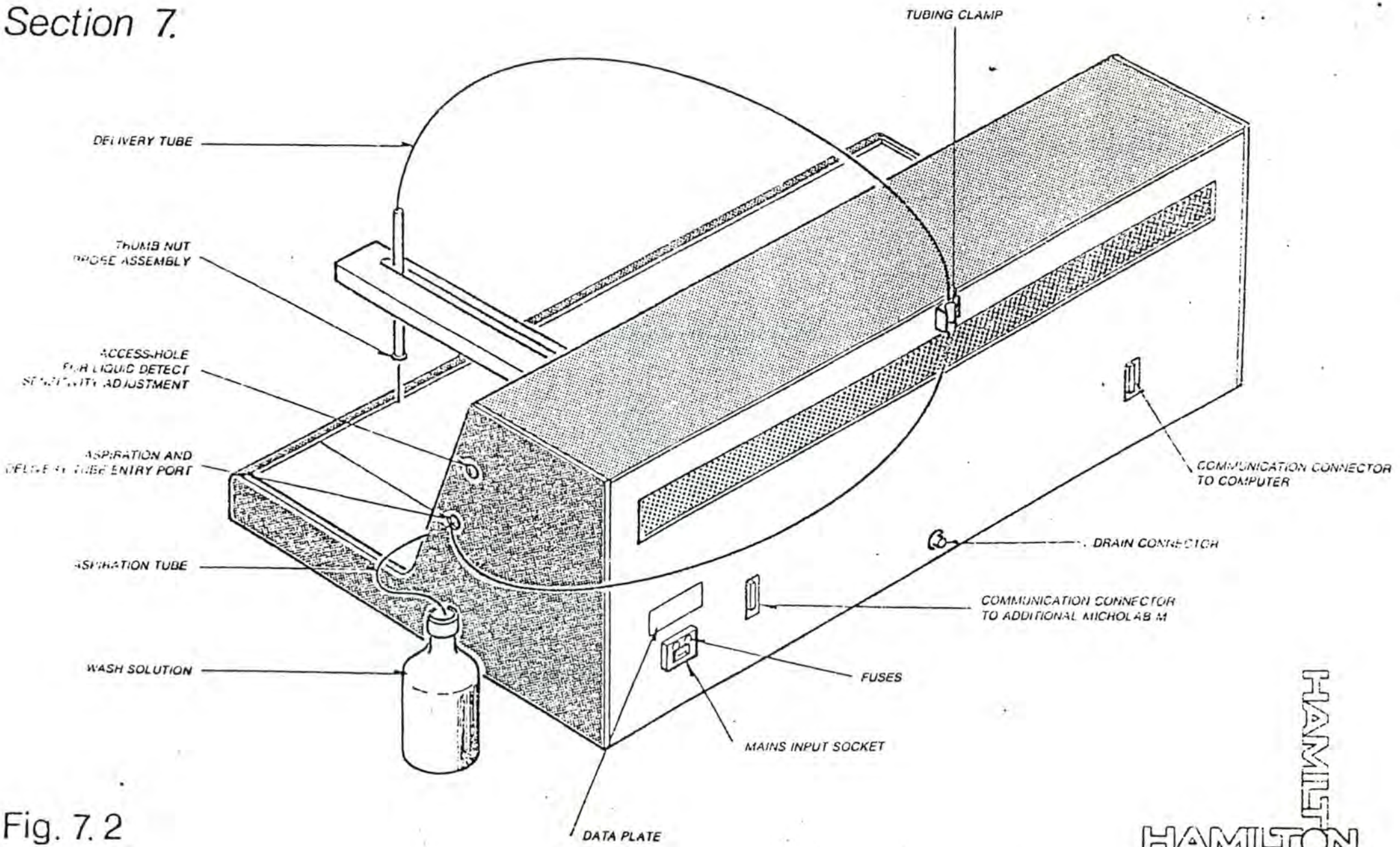
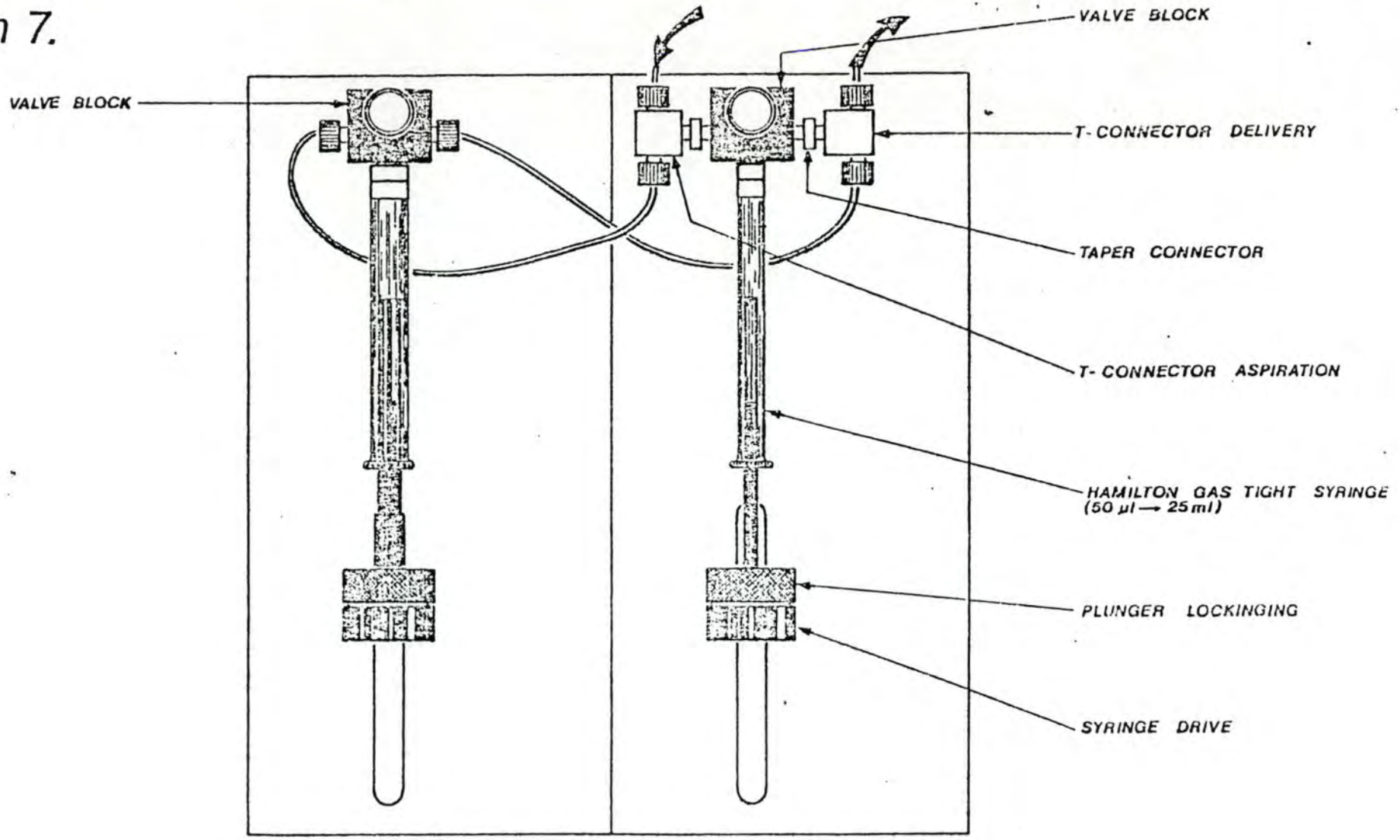


Fig. 7.2

HAMILTON



LIQUID SYSTEM CONNECTIONS

Fig.7. 3

Typical layout

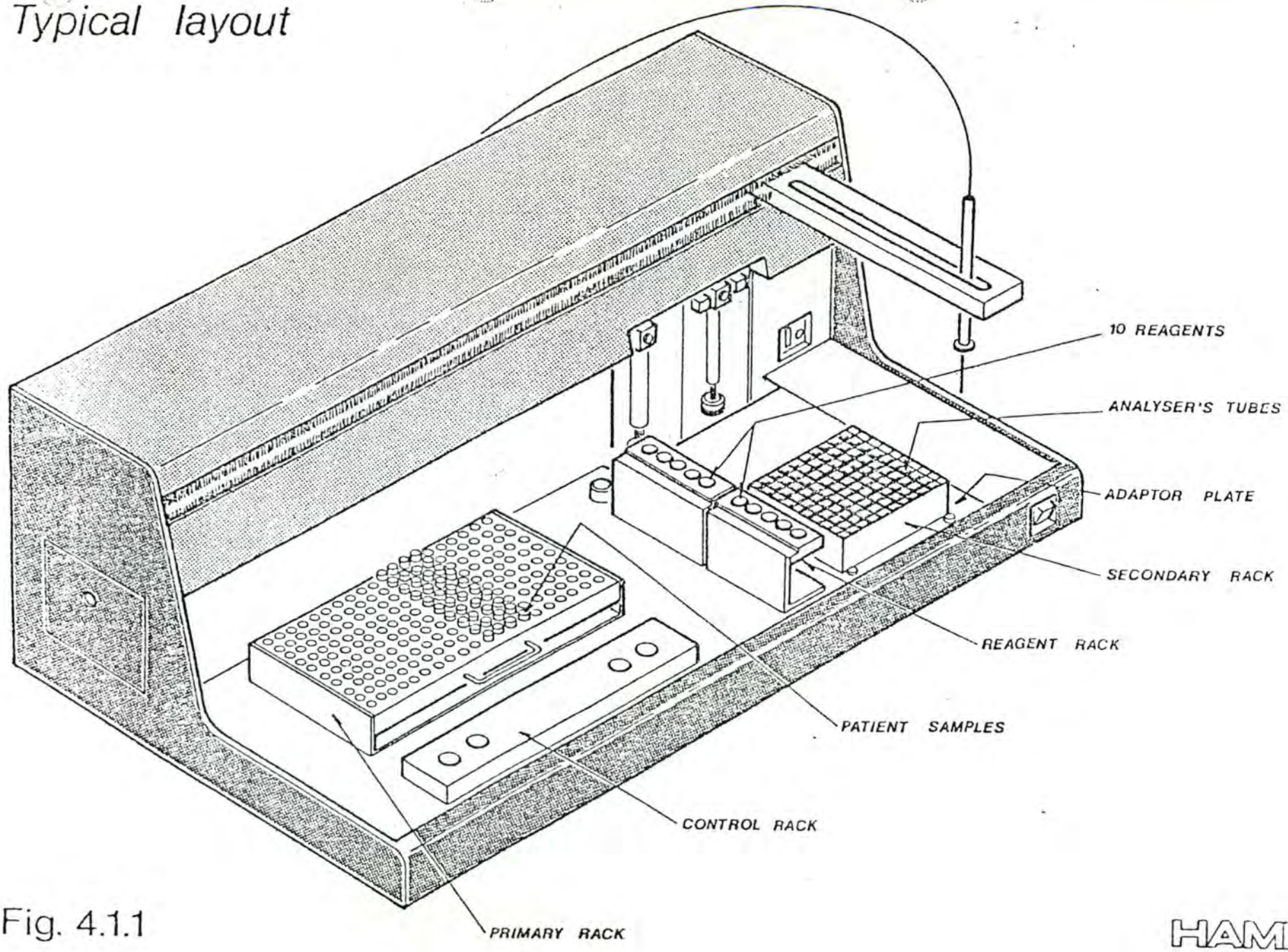
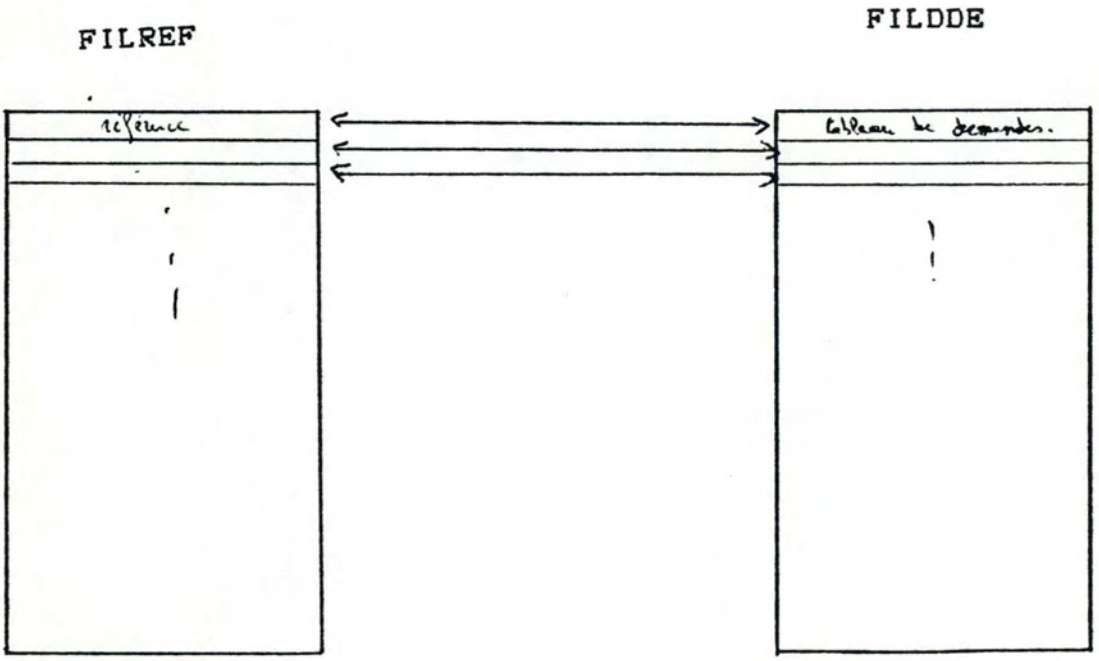


Fig. 4.1.1

HAMILTON

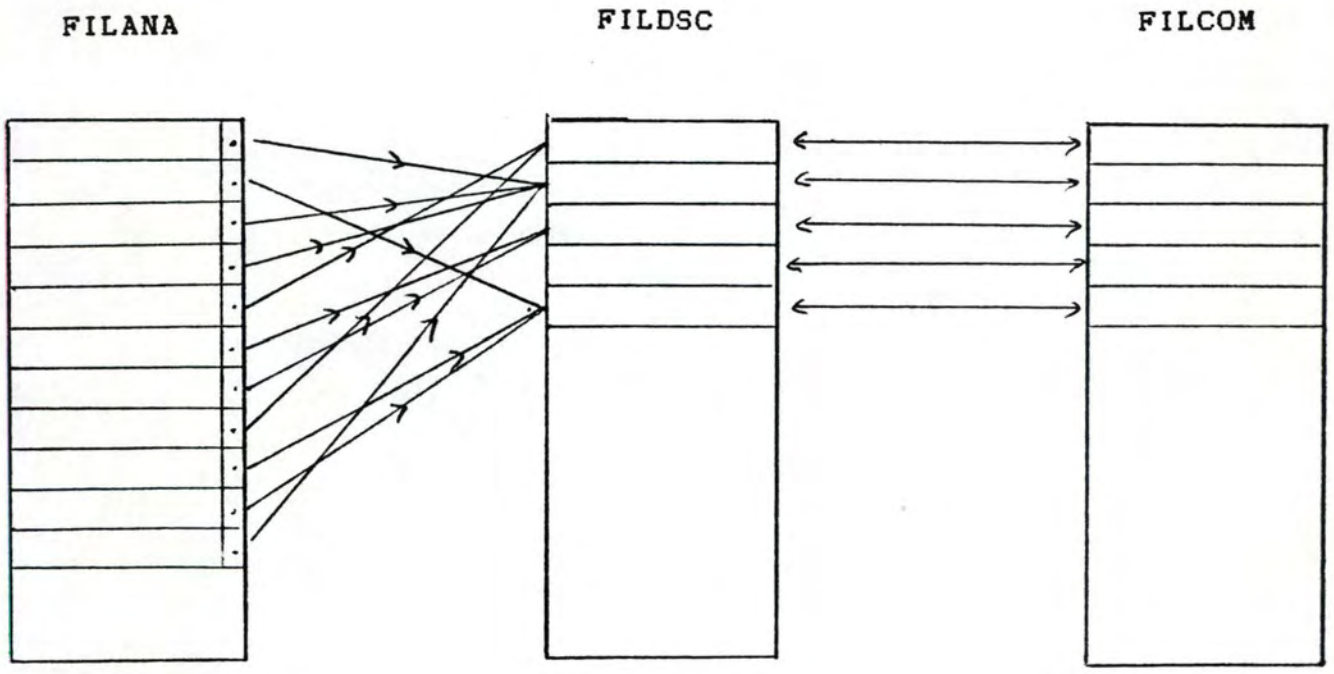
Appendix I

Structure de FILREF et FILDDE.



Appendix J

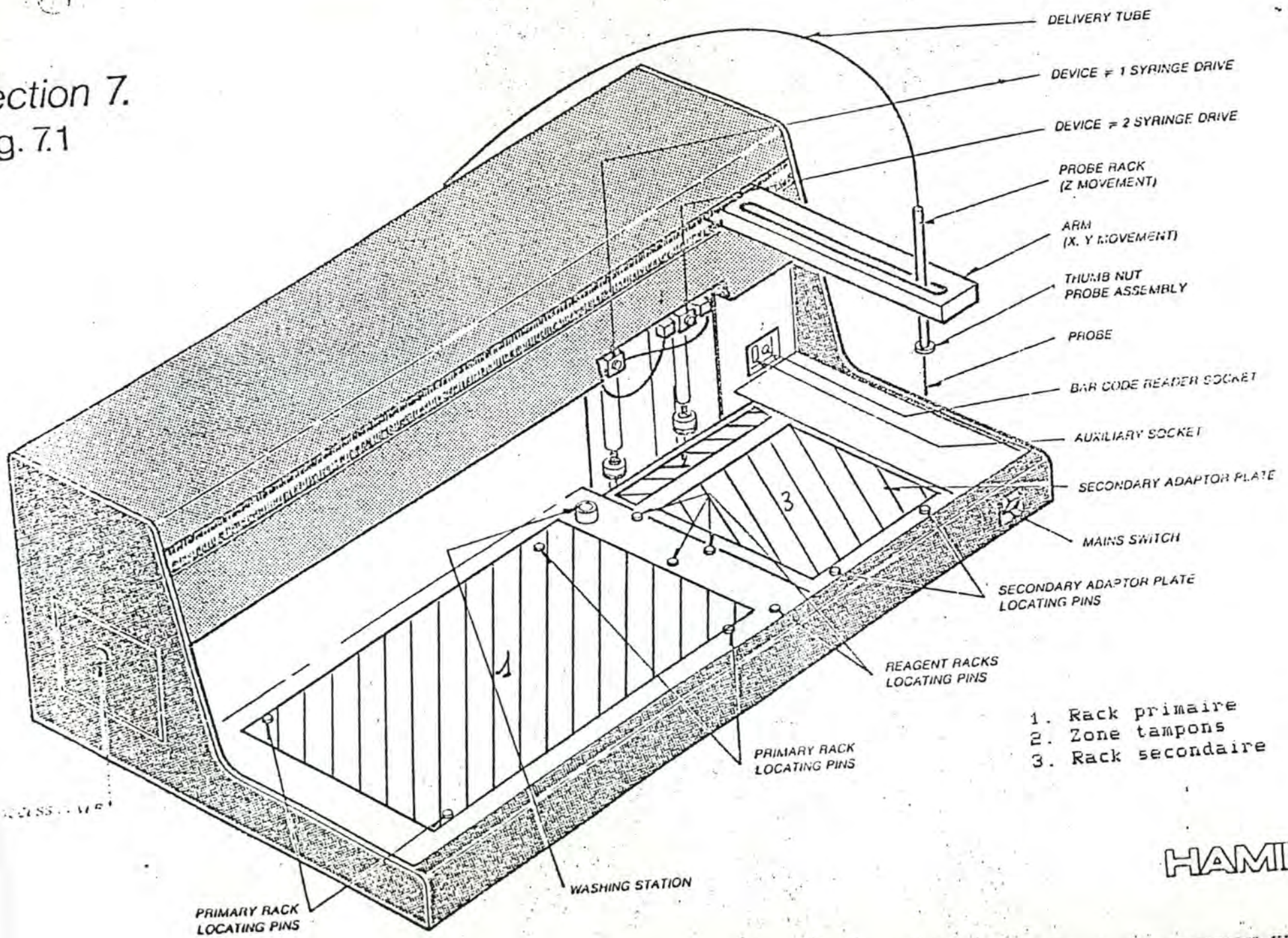
Structure de FILANA, FILDSC et FILCOM.



Appendix K

Table de travail de l'HAMILTON.

Section 7.
Fig. 7.1



1. Rack primaire
2. Zone tampons
3. Rack secondaire

HAMILTON

Appendix L

V52192 Listing LAN12.

```
------
    0 S.VITESSE DE SEDIMENTAT.
    1 S.HEMOGLOBINE
    2 S.GLOBULES ROUGES
    3 S.GLOBULES BLANCS
    4 S.FORMULE LEUCOCYTAIRE
    5 S.HEMATOCRITE
   54 S.GROUPES SANGUINS ABO
   55 S.FACTEUR RH
   66 S.TEMPS DE SAIGNEMENT
   71 S.CEPHALINE (TEMPS DE)
   74 S.PLAQUETTES NUMERATION
  245 S.STREPTOZYME
  255 S.ANTI-PROTEINE M
  258 S.ANTI-HYALURONIDASE
  314 S.PROTHROMBINE ENFANT
  315 S.THROMBOTEST ENFANT
   46 S.ASL (STREPTO)

J59495
------
   96 S.SODIUM
   97 S.POTASSIUM
  102 S.CALCIUM
  105 S.MAGNESIUM SERIQUE
  327 S.MAGNESIUM ERYTHROCYTAI
   35 S.FER

J59549
------
    0 S.VITESSE DE SEDIMENTAT.
    1 S.HEMOGLOBINE
    2 S.GLOBULES ROUGES
    3 S.GLOBULES BLANCS
    4 S.FORMULE LEUCOCYTAIRE
    5 S.HEMATOCRITE
    6 S.UREE
    7 S.GLUCOSE
    8 S.CHOLESTEROL TOTAL
   10 S.LIPIDES TOTAUX
   11 S.LIPIDOGRAMME
   18 S.PROTEINES TOTALES
   19 S.PROTEINES ELECTROPHOR.
   20 S.THYMOL
   23 S.TRIGLYCERIDES
   24 S.KUNKEL ZINC
   26 S.SGOT
```

Index

Table des matières.

Table of Contents

Institut d'Informatique
21, Rue Grandgagnage
B-5000 NAMUR

ANNEE ACADEMIQUE 1983-84

Automatisation du dispatching d'un laboratoire d'analyses médicales

** A N N E X E S **

****

Etienne ALEXANDRE

Promoteur : Mr Jean RAMAEKERS

Mémoire présenté en vue de l'obtention du grade de Licencié et Maitre en Informatique.

Chapter 9

Le probleme du pilotage de l'HAMILTON.

9.1 Introduction.

Pour diverses raisons qui ont été exposée en 3.2, il serait souhaitable de pouvoir piloter l'HAMILTON à partir d'un Xérox qui remplacerait l'actuel ordinateur pilote, le KONTRON.

9.2 Description du système Kontron-Hamilton.

9.2.1 Matériel.

Actuellement le système peut se décrire de la façon suivante:

Microlab M2000 Micro-laboratoire automatique fabriqué par la firme Suisse Hamilton (Cfr appendice H du premier volume). Ce microlab se compose essentiellement de 3 unités (device) (3 autres unités existent en option) dont les mouvements sont commandés par ordinateur.

1. DEVICE #1: Il s'agit d'une micro-pompe fonctionnant comme une seringue permettant d'aspirer ou de débiter des volumes très faibles (min 2.5 micro-litres)

2. DEVICE #2: Il s'agit également d'une micro-pompe semblable à la précédente si ce n'est que le diamètre de son cylindre est un peu plus grand. Elle est donc utilisée pour manipuler des volumes plus importants.

3. DEVICE #3: Cette troisième unité est une aiguille (type seringue) pouvant se déplacer verticalement sur un bras mobile.

L'ensemble constitue donc un élément mobile dans les trois directions X,Y,Z.(Résolution de déplacement en X, Y et Z : environ 0.1 mm)

Computer 6000 Micro-ordinateur Kontron de fabrication allemande (Munich) avec un floppy disk.

- Capacité mémoire centrale : 64 KB
- Capacité d'une disquette : 308 KB

9.2.2 Logiciel.

Les programmes d'application sont écrits en Basic. Les communications entre le Kontron et M2000 sont gérées

- par le module COMMON écrit en Basic qui doit faire partie de chaque programme de pilotage et qui fait appel au module ci-après.
- par le module MLCOMM écrit en assembleur qui constitue le gérant des communications

Ces deux modules constituent ensemble le boîte noire dont il a été question en 3.4

9.2.3 Communications.

Les deux appareils sont reliés entre eux par une jonction RS232 qui permet un dialogue entre les deux machines.

9.3 Fonctionnement du système Kontron-Hamilton.

Au niveau programmation Basic, les ordres (command strings) envoyés au microlab répondent au format suivant:

- #⟨numéro de device⟩⟨command⟩⟨paramètres⟩

exemples :1. #3PA1254,458,321
2. #2IP500R
3. #1Z#3PI

Un command string peut inclure plusieurs ordres (ex 3).

<command> peut être répétitif (ex 2).

<paramètre> est optionnel.

Ces ordres peuvent être de différents types.

- Ordres de positionnement de l'aiguille ou de mouvement des pompes. (positioning commands)

- Ordres de définitions de valeurs de positions.(set commands)

- Ordres de rapport de position ou d'état.(report commands)

Une description plus complète de la syntaxe se trouve en Appendice A.

Les ordres à envoyer sont mis dans une variable de type string "SM$", les réponses du microlab aux "report commands" sont rangées dans une autre variable de type string "RM$", enfin les codes d'erreur sont renvoyés dans un tableau a 6 positions "IEC" (1 élément par device) dont seulement les 3 premiers sont utilisés (device 1,2,3).

La manière "classique" de donner un ordre au microlab est la suivante (Cfr appendice A):

1. assigner à SM$ la chaîne de caractères représentant la commande que doit exécuter le microlab.

2. effectuer GOSUB 50000 , adresse à laquelle se trouve le module COMMON. Celui-ci est un module d'aide au programmeur (tests, progression pas à pas, trace sur imprimante ...) mais se révèle etre assez lourd et lent. A l'intérieur de ce module, on peut trouver l'instruction CALL ZCOMM (SM$,RM$,IEC) qui est un appel à une routine du module assembleur MLCOMM.

3. éventuellement exploiter les valeurs de RM$ et IEC.

Une manière moins classique mais nettement plus rapide consiste à remplacer l'étape 2 par un CALL ZCOMM (SM$,RM$,IEC) direct. Cette méthode est actuellement utilisée au laboratoire et donne de très bons résultats.

L'analyse du gérant des communications (MLCOMM) a permis de découvrir qu'il se compose en fait de 2 routines.

1. ZCINI: routine d'initialisation des ports de sortie. Il en ressort les caractéristiques suivantes:

- vitesse de transmission: 2400 bauds

- 8 bits / caractère

- pas de parité

- 2 stop bits / caractère

2. ZCOMM: routine de gestion des communications. Il s'avère que:

   - ZCOMM transforme les command strings du programme utilisateur afin de leur donner un format correspondant mieux à un protocole de transmission. En particulier, le "#" est remplacé par un autre caractère, le message est encadré des caractères "stx" et "etx" et cloturé par un caractère de contrôle de validité(voir ci-après).

   - Les informations transmises ont donc le format suivant:

     Du Kontron vers l'Hamilton :

     ⟨stx⟩ ⟨alph⟩ ⟨nro dev⟩ ⟨command⟩ ⟨param⟩ ⟨etx⟩ ⟨crc⟩

     ou

     stx = caractère ascii 02, start of text
     alpha = caractère alphabétique :
     A --> message normal
     B --> message répété suite à un problème de liaison
     @ --> le dernier message reçu de l'Hamilton n'a pas été compris(format incorrect ou crc invalide)
     nro dev = numéro du device concerné
     command = suite de caractères de commande valide ou non d'après la syntaxe.
     param = paramètres de ⟨command⟩
     etx = caractère ascii 03, end of text
     crc = caractère de vérification résultant d'un " ou exclusif " des caractères tramsmis, du stx au etx inclus

     De l'Hamilton vers le Kontron:

     ⟨stx⟩ ⟨alpha⟩ ⟨nro dev⟩ ⟨string⟩ ⟨etx⟩ ⟨crc⟩

     ou

     stx = cfr supra.

alpha = caractère alphabétique :

A --> message réponse relatant une erreur, format:
‹stx›"A"‹nro dev› ‹alpha› ‹etx› ‹crc›
ceci signifie en fait que l'élément IEC (nro dev) prendra la valeur ord(alpha)-64 où ord(alpha)représente le numéro d'ordre dans le code ascii du caractère ‹alpha›

C --> message réponse, format 1:
‹stx› "C" ‹nro dev› ‹etx› ‹crc›
signifie que le device ‹nro dev› a terminé d'exécuter une commande qui lui a été envoyée; format 2:
‹stx›"C"‹nro dev› ‹string› ‹etx› ‹crc›
signifie que string est la réponse à une " report command " et se renvoie dans RM#.
ex de ‹string›: "1200" "Y" "N" "0000"

@ --> le dernier message reçu du Kontron n'a pas été compris (format incorrect ou crc invalide)

- Tout command string n'est envoyé à l'Hamilton, que s'il répond aux règles syntaxiques suivantes:

  * Toute commande doit être précédée par "#" suivi du numéro de device concerné.
    Ex: #1IP500R #3PI#20D20R#1Z

  * Une seule commande par device est acceptée dans un command string sauf pour les devices 1 et 2.

  * Une seule "report command" est acceptée dans un command string.

  * Une commande avec l'option "wait" (attendre que le(s) device(s) concerné(s) ai(en)t terminé son(leur) action avant de rendre la main au programme utilisateur) doit être entourée de parenthèses carrées.

    Ex: [#3PI] [#1Z]#2IP500R

- Si le command string contient d'autres erreurs, elles ne sont pas détectées au niveau de ZCOMM. Le command string est envoyé et c'est l'Hamilton qui détecte l'erreur et la signale par un message d'erreur (cfr supra).

9.4 Réécriture de la boîte noire.

9.4.1 Introduction.

Ces routines ont été écrites en PASCAL et assurent pour le Xérox à peu de choses près les mêmes fonctions que le module MLCOMM pour le Kontron.

9.4.2 Spécifications fonctionnelles.

PROCEDURE comm

FONCTION — Se charge d'envoyer une chaîne de caractères à l'Hamilton et de capter une éventuelle réponse.La procédure gère tous les problemes que cela peut impliquer et si l'option wait est positionnée, elle ne rend la main au programme appelant que lorsque le device concerné a signalé par un message qu'il a fini de travailler.

ARGUMENTS

- sm : chaîne de 20 caractères maximun a envoyer.

RESULTATS

- rm : chaîne de 20 caractères maximun contenant éventuellement une réponse de l'Hamilton.
- err : code d'erreur :si = 0 alors la transmission s'est passée sans problèmes.

9.4.3 Texte source.

```
MODULE hardcomm;

CONST
        spdb   = $0C;
        ctrb   = $07;
        datab  = $05;
        kbd    = $1E;
        lmax   = 20 ;
        tempo  = 200;

TYPE
        str20    = string[20]                          ;
        tabdata  = ARRAY [0..lmax] OF INTEGER          ;
        tab3     = ARRAY [1..3]    OF INTEGER          ;

VAR
        encours : EXTERNAL tab3;
        waitopt : EXTERNAL BOOLEAN;
```

```
PROCEDURE resetenc;

VAR x    : INTEGER;

BEGIN
        FOR x:=1 TO 3 DO encours[x]:=0;
END;


PROCEDURE receive (VAR msg:str20;VAR lgr:INTEGER;VAR valid:BOOLEAN);

VAR     crc,crcrec,i,p          : INTEGER;
        tmp                     : tabdata;

BEGIN
        msg:='                  ';
        lgr:= 1;
        p:=0;
        crc := 0;
        tmp[0]:=0;
        OUT[(ctrb)] := $10;
        WHILE ( (lgr < lmax) AND (tmp[lgr-1]<>03) )  DO
        BEGIN
        IF ((INP[(ctrb)] & $1) = 1 ) THEN
                        BEGIN
                                tmp[lgr] := INP[(datab)];
                                IF (tmp[lgr]=2)and(lgr>1)
                                        THEN BEGIN  tmp[1]:=tmp[lgr];
                                                    lgr  := 1;
                                                    crc :=0;
                                                end;
                                crc:= (crc&tmp[lgr]) ! ~(crc!tmp[lgr]);
                                lgr:=lgr+1;
                        END;
        END ;(while)
        WHILE ((INP[(ctrb)] & $1) = 0 ) DO OUT[(ctrb)]:=$10;
        crcrec := INP[(datab)];
        IF crc < 0 THEN BEGIN   crc :=crc & $7F;
                                crc:= ~crc;
                                crc :=crc & $7F;
                        END;
        IF crc <> crcrec THEN valid:=FALSE
                         ELSE BEGIN valid := TRUE;
                                    FOR i:=2 TO lgr-2 DO
                                      msg[i-1]:=chr(tmp[i]);
                                      lgr:=lgr-3;
                                END;
END;
```

```
PROCEDURE send (msg :str20);

VAR     i,crc,x,lgr     :INTEGER;
        tmp             :tabdata;

BEGIN
        lgr   := LENGTH (msg);
        tmp[1]     :=02;
        for i := 2 to lgr+1 do tmp[i]:=ord(msg[i-1]);
        tmp[lgr+2] :=03;
        crc        :=0;
        FOR i:= 1 TO lgr+2 do crc:= (crc&tmp[i]) ! ~(crc!tmp[i]);
        lgr         :=lgr+3;
        IF crc < 0 THEN BEGIN   crc :=crc & $7F;
                                crc:= ~crc;
                                crc :=crc & $7F;
                        END;
        tmp[lgr]   :=crc;
        FOR i:=1 TO lgr DO
        BEGIN
                WHILE ((INP[(ctrb)] & $4)=1) DO OUT[(ctrb)]:=$10;
                OUT[(datab)]:=tmp[i];
                FOR x:=1 TO tempo DO;
        END;
END;
```

```
PROCEDURE comm (sm:str20;VAR rm :str20;VAR err:INTEGER);

LABEL 55;

VAR     i,lgr              : INTEGER;
        tmpsend,msg        : str20  ;
        valid              : BOOLEAN;


PROCEDURE trtmsg;

VAR     i                  : INTEGER;
        must               : BOOLEAN;
BEGIN
        must := FALSE;
        IF lgr >1
          THEN encours[ORD(msg[2])-48]:=0;{encours[ORD(msg[2])-48]-1;}
        CASE lgr of
        1: CASE msg[1] OF
                '@' : GOTO 55;
                'A' : must := TRUE;
                ELSE WRITELN('IMPREVU 1 >',msg[1],'<');
           END;

        2: CASE msg[1] OF
                'C' : ;
                ELSE WRITELN('IMPREVU 2 >',msg[1],'<');
           END;
        3: CASE msg[1] OF
                'C' : rm:=msg[3];
                'A' : err:= ord(msg[3])-64;
                ELSE BEGIN
                        WRITELN('IMPREVU 3 >',msg[1],'<');
                        err:= ord(msg[3])-64;
                     END;
           END;
        ELSE  FOR i :=1 TO lgr-2 do rm[i]:=msg[i+2];
        END;
        IF ( (encours[ORD(sm[1])-48] <>0) AND (waitopt) ) OR must
                THEN send('A');
END;
```

```
(PROG PRINC)
BEGIN
        err  := 0;
        rm :='                          ';
        encours[ORD(sm[1])-48] := encours[ORD(sm[1])-48] +1 ;
55:     send(CONCAT('A',sm));
        REPEAT
        RECEIVE (msg,lgr,valid);
        WHILE NOT valid DO
        BEGIN
                send('@');
                RECEIVE (msg,lgr,valid);
                WRITELN('INVALID MSG');
        END;
        trtmsg;
        UNTIL ( (encours[ORD(sm[1])-48] =0) OR (not waitopt) );

END;

MODEND.
```

9.4.4 Déclaration.

```
PROCEDURE comm (sm:str20;VAR rm :str20;VAR err:INTEGER);
```

9.5 Développement d'un logiciel d'aide a la mise au point en PASCAL de programmes de pilotage de l'Hamilton.

9.5.1 Introduction

La conception d'un programme PASCAL est assez différente de celle d'un programme en BASIC et il m'a semblé utile d'écrire quelques procédures et fonctions permettant d'écrire des programmes sans devoir descendre au niveau de la manipulation de chaînes de caractères de commande comme c'était le cas en BASIC.

L'utilisation de ces procédures devra permettre à l'utilisateur

1. De ne pas s'occuper de la gestion des communinations.
2. De ne plus s'embarasser de la construction et de l'assemblage de chaînes de caractères conformément a une syntaxe donnée.

Par exemple, la procédure PASCAL

```
PINIXYZ
```

pourrait remplacer l'instruction BASIC

```
CALL ZCOMM('#3PI',RM$,IEC).
```

9.5.2 Logiciel de base.

9.5.2.1 Introduction.

Ce logiciel de base comprend en fait une série de procédures reproduisant exactement les commandes fondamentales (cfr Append. A) que l'on peut envoyer à l'Hamilton. Il s'agit donc de procedures de très bas niveau. Ces procédures sont indispensables pour 2 raisons:

1. comme elles assurent la même fonction qu'une instruction BASIC suivie d'un appel à la boîte noire, elle permettent une réécriture rapide des programmes BASIC existants en PASCAL.

2. elles vont servir de base à l'élaboration de procédures plus "évoluées" et permettront de plus à l'utilisateur, dont l'application ne se satisferait pas du logiciel étendu développé ci-après, de construire son propre niveau supérieur de logiciel.

9.5.2.2 Spécifications fonctionnellles.

PROCEDURE setwopt

FONCTION — Positionne à "oui" l'option wait, c-à-d que le programme utilisateur sera bloqué tant que l'action ordonnée à l'Hamilton ne sera pas terminée. L'option est positionnée à "non" lors de initrsm.

ARGUMENTS

- néant

RESULTATS

- néant

PROCEDURE resetwopt

FONCTION — Positionne à "non" l'option wait, c-à-d que le programme utilisateur peut s'exécuter pendant les mouvements de l'Hamilton.

ARGUMENTS

- néant

RESULTATS

- néant

PROCEDURE initrsm

FONCTION — Initialise le processus de transmission ainsi que les données globales du programme. Cette procédure doit OBLIGATOIREMENT être invoquée dans le programme avant de faire quelqu'action que ce soit.

ARGUMENTS

- néant

RESULTATS

- néant

PROCEDURE pinixyz

FONCTION Positionne l'aiguille en x=0,y=0,z=0 (position initiale).

ARGUMENTS

- néant

RESULTATS

- néant

PROCEDURE piniz

FONCTION Positionne l'aiguille en z=0 en laissant x et y inchangés (z-position initiale).

ARGUMENTS

- néant

RESULTATS

- néant

PROCEDURE princage

FONCTION Positionne l'aiguille dans la partie centrale (remplie d'eau) de la station de lavage afin de pouvoir rincer l'extérieur de l'aiguille.

ARGUMENTS

- néant

RESULTATS

- néant

PROCEDURE pvidange

FONCTION Positionne l'aiguille dans la partie extérieure (reliée a un bac de vidange) de la station de lavage afin de pouvoir y jeter du liquide inutile.

ARGUMENTS

- néant

RESULTATS

- néant

PROCEDURE pabs

FONCTION Positionne l'aiguille en position absolue x,y,z.

ARGUMENTS

- x: x-position désirée
- y: y-position désirée
- z: z-position désirée

RESULTATS

- err: code d'erreur

PROCEDURE pzabs

FONCTION Positionne l'aiguille en z-position absolue z en laissant inchangés x et y.

ARGUMENTS

- z: z-position désirée

RESULTATS

- err: code d'erreur

PROCEDURE pzrel

FONCTION Effectue un déplacement relatif de l'aiguille de +/- n pas.

ARGUMENTS

- npas: nombre de pas du déplacement +: vers le bas ; -: vers le haut

RESULTATS

- err: code d'erreur

PROCEDURE pzdip

FONCTION Positionne l'aiguille en zd-position en laissant inchangés x et y.

ARGUMENTS

- neant

RESULTATS

- err: code d'erreur

PROCEDURE submerge

FONCTION Fait descendre l'aiguille jusqu'au liquide y pénètre de n pas.

ARGUMENTS

- npas: nombre de pas vers le bas que l'aiguille doit effectuer une fois le liquide détecté

RESULTATS

- err: code d'erreur

PROCEDURE defzpos

FONCTION Définit les 3 z-positions caractéristiques zm,zc,zd.

ARGUMENTS

- zm: cfr Append. A.3
- zc: cfr Append. A.3
- zd: cfr Append. A.3

RESULTATS

- err: code d'erreur

PROCEDURE defzvdg

FONCTION Définit la z-position de vidange c-à-d la hauteur à laquelle va se positionner l'aiguille lors d'un appel à pvidange.

ARGUMENTS

- z: z-position pour la vidange.

RESULTATS

- err: code d'erreur

PROCEDURE defzrcg

FONCTION Définit la z-position de rincage c-à-d la hauteur à laquelle va se positionner l'aiguille lors d'un appel à princage.

ARGUMENTS

- z: z-position pour le rincage.

RESULTATS

- err: code d'erreur

PROCEDURE defzvdg

FONCTION Définit la z-position pour le prochain mouvement en x et y c-a-d la hauteur à laquelle va rester l'aiguille pendant le prochain déplacement.

ARGUMENTS

- z: z-position de déplacement.

RESULTATS

- err: code d'erreur

PROCEDURE repmzpos

FONCTION Fournit la z-position maximale assignable (hauteur jusqu'à laquelle on peut descendre) étant donné la position en x et y courante.

ARGUMENTS

- néant

RESULTATS

- mzpos: z-position maximale assignable

PROCEDURE repxpos

FONCTION Fournit la position actuelle en x.

ARGUMENTS

- néant

RESULTATS

- xpos: x-position courante.

PROCEDURE repypos

FONCTION Fournit la position actuelle en y.

ARGUMENTS

- néant

RESULTATS

- ypos: y-position courante.

PROCEDURE repzpos

FONCTION Fournit la position actuelle en z.

ARGUMENTS

- néant

RESULTATS

- zpos: z-position courante.

PROCEDURE pmpout

FONCTION Aspirer de "npas" en output avec la pompe numéro "npmp".

ARGUMENTS

- npmp: numéro de pompe.
- npas: nombre de pas.

RESULTATS

- err: code d'erreur

PROCEDURE pmpin

FONCTION Aspirer de "npas" en input avec la pompe numéro "npmp".

ARGUMENTS

- npmp: numéro de pompe.
- npas: nombre de pas.

RESULTATS

- err: code d'erreur

PROCEDURE refout

FONCTION Dévider de "npas" en output avec la pompe numéro "npmp".

ARGUMENTS

- npmp: numéro de pompe.
- npas: nombre de pas.

RESULTATS

- err: code d'erreur

PROCEDURE defvitpl

FONCTION Définit la vitesse de plongée.

ARGUMENTS

- npmp: numéro de pompe.
- vit: vitesse (0-15)

RESULTATS

- err: code d'erreur

PROCEDURE descdstop

FONCTION Descendre de nn avant de s'arrêter

ARGUMENTS

- npmp: numéro de pompe.
- npas: nombre de pas.(0-99)

RESULTATS

- err: code d'erreur

PROCEDURE pmpini

FONCTION Initialise la pompe numéro "npmp".

ARGUMENTS

- npmp: numéro de pompe.

RESULTATS

- err: code d'erreur

PROCEDURE pmptest

FONCTION Teste la pompe numéro "npmp".

ARGUMENTS

- npmp: numéro de pompe.

RESULTATS

- ready: TRUE si la pompe est OK, FALSE sinon.

!!!!! **AVERTISSEMENT** !!!!!

Il est fortement déconseillé d'envoyer plusieurs ordres successifs à un même device sans positionner l'option wait car les résultats risquent d'être très inattendus et catastrophiques.

9.5.2.3 Texte source.

```
MODULE softcomm;

TYPE    STR4     = STRING[4]      ;
        STR20    = STRING[20]     ;
        tab3     = ARRAY [1..3]     OF INTEGER ;


VAR
        encours          : tab3          ;
        waitopt          : BOOLEAN       ;
        xposcour         : INTEGER       ;
        yposcour         : INTEGER       ;
        zposcour         : INTEGER       ;


EXTERNAL PROCEDURE resetenc;
EXTERNAL PROCEDURE bsiopen;
EXTERNAL PROCEDURE comm
         (message:str20;VAR reponse:str20;VAR coderr:integer);

EXTERNAL FUNCTION POWER ( nb : integer; puiss :integer):integer;
```

```
PROCEDURE newval;
VAR     wtmp              : BOOLEAN       ;
BEGIN   wtmp    := waitopt                          ;
        setwopt                                     ;
        repxpos(xposcour)                           ;
        repypos(yposcour)                           ;
        repzpos(zposcour)                           ;
        waitopt := wtmp                             ;
END;
```

```
PROCEDURE cvint           (str:str4; VAR n:INTEGER);
VAR     i,lg    :INTEGER;
BEGIN
        n := 0;
        lg := LENGTH(str);
        IF lg = 0
        THEN n := 9999
        ELSE FOR i := 1 TO lg DO n := n * 10 + ORD(str[i])-48;
END;
```

```
PROCEDURE cvstr           (n:INTEGER; VAR str:str4);
VAR     a,i,p   :INTEGER;
BEGIN
        str     := '0000' ;
        IF (n > 9999) or (n<0)
        THEN str := '9999'
        ELSE
        FOR i := 3 DOWNTO 0 DO
        BEGIN
                p := power(10,i);
                a := n DIV p;
                n := n MOD p;
                str [4-i] := chr(48+a);
        END;
END;
```

```
PROCEDURE pinixyz ;
VAR     msg,rep          : str20          ;
        cderr            : INTEGER        ;
BEGIN
        xposcour:= 0                                ;
        yposcour:= 0                                ;
        zposcour:= 0                                ;
        msg      := '3PI'                           ;
        comm     (msg,rep,cderr)                    ;
        IF cderr <> 0 then newval                   ;
END;



PROCEDURE piniz ;
VAR     msg,rep          : str20          ;
        cderr            : INTEGER        ;
BEGIN
        zposcour:= 0                                ;
        msg      := '3ZI'                           ;
        comm     (msg,rep,cderr)                    ;
        IF cderr <> 0 then newval                   ;
END;



PROCEDURE pabs           (x,y,z:INTEGER; VAR err:INTEGER);
VAR     msg,rep          : str20          ;
        cderr            : INTEGER        ;
        strx,stry,strz   : str4           ;
BEGIN
        xposcour:= x                                ;
        yposcour:= y                                ;
        zposcour:= z                                ;
        cvstr    (x,strx)                           ;
        cvstr    (y,stry)                           ;
        cvstr    (z,strz)                           ;
        msg      :='                    '           ;
        msg      := concat('3PA',strx,',',stry,',',strz);
        comm     (msg,rep,cderr)                    ;
        err      := cderr                           ;
        IF cderr <> 0 then newval                   ;
END;
```

```
PROCEDURE princage ;
VAR     msg,rep          : str20          ;
        cderr            : INTEGER        ;
BEGIN
        msg      := '3PC'                         ;
        comm     (msg,rep,cderr)                  ;
        newval                                    ;
END;



PROCEDURE pvidange ;
VAR     msg,rep          : str20          ;
        cderr            : INTEGER        ;
        strx,stry,strz   : str4           ;
BEGIN
        msg      := '3PW'                         ;
        comm     (msg,rep,cderr)                  ;
        newval                                    ;
END;



PROCEDURE pzabs          (zposabs:INTEGER; VAR err:INTEGER);
VAR     msg,rep          : str20          ;
        cderr            : INTEGER        ;
        strz             : str4           ;
BEGIN
        zposcour:= zposabs                        ;
        cvstr    (zposabs,strz)                   ;
        msg      :='      '                       ;
        msg      := concat('3ZA',strz)            ;
        comm     (msg,rep,cderr)                  ;
        err      := cderr                         ;
        IF cderr <> 0 then newval                 ;
END;
```

```
PROCEDURE pzrel          (zposrel:INTEGER; VAR err : INTEGER);
VAR     msg,rep          : str20          ;
        cderr            : INTEGER        ;
        strz             : str4           ;
        signe            : CHAR           ;
BEGIN
        zposcour:= zposcour + zposrel                    ;
        IF zposrel <0
        THEN BEGIN
                signe := '-'                             ;
                zposrel := - zposrel                     ;
             END
        ELSE    signe := '+'                             ;
        cvstr   (zposrel,strz)                           ;
        msg     := '        '                            ;
        msg     := concat('3ZR',signe,strz)              ;
        comm    (msg,rep,cderr)                          ;
        err     := cderr                                 ;
        IF cderr <> 0 then newval                        ;
END;
```

```
PROCEDURE pzdip          (VAR err:INTEGER);
VAR     msg,rep          : str20          ;
        cderr            : INTEGER        ;
        wtmp             : BOOLEAN        ;
BEGIN
        msg     := '3ZD'                                 ;
        comm    (msg,rep,cderr)                          ;
        err     := cderr                                 ;
        wtmp    := waitopt                               ;
        setwopt                                          ;
        repzpos (zposcour)                               ;
        waitopt := wtmp                                  ;
        IF cderr <> 0 then newval                        ;
END;
```

```
PROCEDURE submerge      (npas:INTEGER; VAR err:INTEGER);
VAR     msg,rep         : str20         ;
        cderr           : INTEGER       ;
        strpas          : str4          ;
        wtmp            : BOOLEAN       ;
BEGIN
        cvstr   (npas,strpas)                   ;
        msg     := '        '                   ;
        msg     := concat('3ZX',strpas)         ;
        comm    (msg,rep,cderr)                 ;
        err     := cderr                        ;
        wtmp    := waitopt                      ;
        setwopt                                 ;
        repzpos (zposcour)                      ;
        waitopt := wtmp                         ;
        IF cderr <> 0 then newval               ;
END;


PROCEDURE defzpos       (zm,zc,zd:INTEGER;VAR err : INTEGER);
VAR     msg,rep         : str20         ;
        cderr           : INTEGER       ;
        strm,strc,strd  : str4          ;
BEGIN
        cvstr   (zm,strm)                       ;
        cvstr   (zc,strc)                       ;
        cvstr   (zd,strd)                       ;
        msg     := '                    '       ;
        msg     := concat('3SA',strm,',',strc,',',strd);
        comm    (msg,rep,cderr)                 ;
        err     := cderr                        ;
END;


PROCEDURE defzvdg       (zpos:INTEGER ; VAR err:INTEGER);
VAR     msg,rep         : str20         ;
        cderr           : INTEGER       ;
        strz            : str4          ;
BEGIN
        cvstr   (zpos,strz)                     ;
        msg     := '        '                   ;
        msg     := concat('3SW',strz)           ;
        comm    (msg,rep,cderr)                 ;
        err     := cderr                        ;
END;
```

```
PROCEDURE defzrcg       (zpos:INTEGER; VAR err:INTEGER);
VAR     msg,rep         : str20         ;
        cderr           : INTEGER       ;
        strz            : str4          ;
BEGIN
        cvstr   (zpos,strz)                     ;
        msg     := '      '                     ;
        msg     := concat('3SC',strz)           ;
        comm    (msg,rep,cderr)                 ;
        err     := cderr                        ;
END;
```

```
PROCEDURE defzmvt       (zpos:INTEGER; VAR err:INTEGER);
VAR     msg,rep         : str20         ;
        cderr           : INTEGER       ;
        strz            : str4          ;
BEGIN
        cvstr   (zpos,strz)                     ;
        msg     := '      '                     ;
        msg     := concat('3SZ',strz)           ;
        comm    (msg,rep,cderr)                 ;
        err     := cderr                        ;
END;
```

```
PROCEDURE repmzpos      (VAR mzpos:INTEGER);
VAR     msg,rep         : str20         ;
        cderr           : INTEGER       ;
        wtmp            : BOOLEAN       ;
BEGIN
        wtmp    := waitopt                      ;
        setwopt                                 ;
        msg     := '3RM'                        ;
        comm    (msg,rep,cderr)                 ;
        cvint   (rep,mzpos)                     ;
        waitopt := wtmp                         ;
END;
```

```
PROCEDURE repxpos          (VAR xpos:INTEGER);
VAR     msg,rep            : str20          ;
        cderr              : INTEGER        ;
        wtmp               : BOOLEAN        ;
BEGIN
        wtmp     := waitopt                  ;
        setwopt                              ;
        msg      := '3RX'                    ;
        comm     (msg,rep,cderr)             ;
        cvint    (rep,xpos)                  ;
        waitopt  := wtmp                     ;
END;
```

```
PROCEDURE repypos          (VAR ypos:INTEGER);
VAR     msg,rep            : str20          ;
        cderr              : INTEGER        ;
        wtmp               : BOOLEAN        ;
BEGIN
        wtmp     := waitopt                  ;
        setwopt                              ;
        msg      := '3RY'                    ;
        comm     (msg,rep,cderr)             ;
        cvint    (rep,ypos)                  ;
        waitopt  := wtmp                     ;
END;
```

```
PROCEDURE repzpos          (VAR zpos:INTEGER);
VAR     msg,rep            : str20          ;
        cderr              : INTEGER        ;
        wtmp               : BOOLEAN        ;
BEGIN
        wtmp     := waitopt                  ;
        setwopt                              ;
        msg      := '3RZ'                    ;
        comm     (msg,rep,cderr)             ;
        cvint    (rep,zpos)                  ;
        waitopt  := wtmp                     ;
END;
```

```
PROCEDURE pmpout          (npmp,npas:INTEGER; VAR err:INTEGER);
VAR     msg,rep           : str20          ;
        cderr             : INTEGER        ;
        strpas            : str4           ;
        nropmp            : CHAR           ;
BEGIN
        IF npmp = 1 THEN nropmp :='1' ELSE nropmp :='2' ;
        cvstr   (npas,strpas)                          ;
        msg     := '        '                          ;
        msg     := concat(nropmp,'OP',strpas,'R');
        comm    (msg,rep,cderr)                        ;
        err     := cderr                               ;
END;
```

```
PROCEDURE pmpin           (npmp,npas:INTEGER; VAR err:INTEGER);
VAR     msg,rep           : str20          ;
        cderr             : INTEGER        ;
        strpas            : str4           ;
        nropmp            : CHAR           ;
BEGIN
        IF npmp = 1 THEN nropmp :='1' ELSE nropmp :='2' ;
        cvstr   (npas,strpas)                          ;
        msg     := '        '                          ;
        msg     := concat(nropmp,'IP',strpas,'R');
        comm    (msg,rep,cderr)                        ;
        err     := cderr                               ;
END;
```

```
PROCEDURE refout          (npmp,npas:INTEGER; VAR err:INTEGER);
VAR     msg,rep           : str20          ;
        cderr             : INTEGER        ;
        strpas            : str4           ;
        nropmp            : CHAR           ;
BEGIN
        IF npmp = 1 THEN nropmp :='1' ELSE nropmp :='2' ;
        cvstr   (npas,strpas)                          ;
        msg     := '        '                          ;
        msg     := concat(nropmp,'OD',strpas,'R');
        comm    (msg,rep,cderr)                        ;
        err     := cderr                               ;
END;
```

```
PROCEDURE defvitpl      (npmp,vit:INTEGER;VAR err:INTEGER);
VAR     msg,rep         : str20         ;
        cderr           : INTEGER       ;
        strvit          : str4          ;
        nropmp          : CHAR          ;
BEGIN

        IF npmp = 1 THEN nropmp :='1' ELSE nropmp :='2' ;
        cvstr   (vit,strvit)                        ;
        msg     := '      '                         ;
        msg     := concat(nropmp,'S',strvit,'R');
        comm    (msg,rep,cderr)                     ;
        err     := cderr                            ;
END;



PROCEDURE descdstop     (npmp,npas:INTEGER;VAR err:INTEGER);
VAR     msg,rep         : str20         ;
        cderr           : INTEGER       ;
        strpas          : str4          ;
        nropmp          : CHAR          ;
BEGIN

        IF npmp = 1 THEN nropmp :='1' ELSE nropmp :='2' ;
        cvstr   (npas,strpas)                       ;
        msg     := '      '                         ;
        msg     := concat(nropmp,'L',strpas,'R');
        comm    (msg,rep,cderr)                     ;
        err     := cderr                            ;
END;



PROCEDURE pmpini        (npmp:INTEGER);
VAR     msg,rep         : str20         ;
        cderr           : INTEGER       ;
        nropmp          : CHAR          ;
BEGIN
        IF npmp = 1 THEN nropmp :='1' ELSE nropmp :='2' ;
        msg     := '    '                           ;
        msg     := concat(nropmp,'ZR')              ;
        comm    (msg,rep,cderr)                     ;
END;
```

```
PROCEDURE pmptest        (npmp:INTEGER;VAR ready:BOOLEAN);
VAR     msg,rep          : str20          ;
        cderr            : INTEGER        ;
        nropmp           : CHAR           ;
        wtmp             : BOOLEAN        ;
BEGIN
        wtmp     := waitopt                                    ;
        setwopt                                                ;
        IF npmp = 1 THEN nropmp :='1' ELSE nropmp :='2'        ;
        msg      := '   '                                     ;
        msg      := concat(nropmp,'RS')                        ;
        comm     (msg,rep,cderr)                               ;
        IF rep='Y' THEN ready := TRUE ELSE ready := FALSE;
        waitopt := wtmp                                        ;
END;



PROCEDURE setwopt ;
BEGIN
        waitopt := TRUE;
END;



PROCEDURE resetwopt ;
BEGIN
        waitopt := FALSE;
END;



PROCEDURE initrsm ;
BEGIN
        bsiopen          ;
        resetenc         ;
        newval           ;
END;

MODEND.
```

9.5.2.4 Déclarations.

```
PROCEDURE setwopt                                                    ;
PROCEDURE resetwopt                                                  ;
PROCEDURE initrsm                                                    ;
PROCEDURE pinixyz                                                    ;
PROCEDURE piniz                                                      ;
PROCEDURE princage                                                   ;
PROCEDURE pvidange                                                   ;
PROCEDURE pabs       (x,y,z:INTEGER;      VAR err:INTEGER);
PROCEDURE pzabs      (zposabs:INTEGER;    VAR err:INTEGER);
PROCEDURE pzrel      (zposrel:INTEGER;    VAR err:INTEGER);
PROCEDURE pzdip      (                    VAR err:INTEGER);
PROCEDURE submerge   (npas:INTEGER;       VAR err:INTEGER);
PROCEDURE defzpos    (zm,zc,zd:INTEGER;   VAR err:INTEGER);
PROCEDURE defzvdg    (zpos:INTEGER ;      VAR err:INTEGER);
PROCEDURE defzrcg    (zpos:INTEGER;       VAR err:INTEGER);
PROCEDURE defzmvt    (zpos:INTEGER;       VAR err:INTEGER);
PROCEDURE repmzpos   (VAR mzpos:INTEGER)                             ;
PROCEDURE repxpos    (VAR xpos:INTEGER)                              ;
PROCEDURE repypos    (VAR ypos:INTEGER)                              ;
PROCEDURE repzpos    (VAR zpos:INTEGER)                              ;
PROCEDURE pmpout     (npmp,npas:INTEGER; VAR err:INTEGER);
PROCEDURE pmpin      (npmp,npas:INTEGER; VAR err:INTEGER);
PROCEDURE refout     (npmp,npas:INTEGER; VAR err:INTEGER);
PROCEDURE defvitpl   (npmp,vit:INTEGER;  VAR err:INTEGER);
PROCEDURE descdstop  (npmp,npas:INTEGER; VAR err:INTEGER);
PROCEDURE pmpini     (npmp:INTEGER)                                  ;
PROCEDURE pmptest    (npmp:INTEGER;     VAR ready:BOOLEAN);
```

9.5.3 Logiciel étendu.

9.5.3.1 Introduction.

Il s'agit ici de procédures de niveau supérieur écrites dans le cadre specifique du développement d'un programme de dispatching. Ceci n'exclut toutefois pas le fait que certaines de ces procédures puissent être réutilisées dans d'autres contextes.

9.5.3.2 Spécifications fonctionnellles.

PROCEDURE pxrel

FONCTION Effectue un déplacement relatif en x de l'aiguille de +/- n pas.

ARGUMENTS

- npas: nombre de pas du déplacement

RESULTATS

- err: code d'erreur

PROCEDURE pyrel

FONCTION Effectue un déplacement relatif en y de l'aiguille de +/- n pas.

ARGUMENTS

- npas: nombre de pas du déplacement

RESULTATS

- err: code d'erreur

PROCEDURE prackd

FONCTION Se positionne à l'endroit désiré du rack de droite utilisé pour le dispatching.

ARGUMENTS

- nrox : numéro de la colonne (1 -> 19)
- nroy : numéro de la ligne (1 -> 12)
- h : hauteur de positionnement.
- hmvt : hauteur de déplacement.

RESULTATS

- néant

PROCEDURE prackqc

FONCTION Se positionne à l'endroit désiré du rack de gauche (en quinconce) utilisé pour le dispatching.

ARGUMENTS

- position : numéro de l'emplacement désiré (1 -> 500).
- h : hauteur de positionnement.
- hmvt : hauteur de déplacement.

RESULTATS

- néant

PROCEDURE check

FONCTION Effectue l'initialisation des 3 devices de l'Hamilton,càd

- positionne le bras en 0,0,0.
- nettoie,vide et amorce les 2 pompes

ARGUMENTS

- néant

RESULTATS

- néant

9.5.3.3 Texte source.

```
MODULE extdcomm;

CONST   maxxpos = 5365;
        maxypos = 2360;
        maxzpos = 1800;


TYPE    STR4    = STRING[4]     ;
        STR20   = STRING[20]    ;
        tab3    = ARRAY [1..3]    OF INTEGER ;

VAR
        encours         : EXTERNAL tab3         ;
        waitopt         : EXTERNAL BOOLEAN      ;
        xposcour        : EXTERNAL INTEGER      ;
        yposcour        : EXTERNAL INTEGER      ;
        zposcour        : EXTERNAL INTEGER      ;
```

```
declaration des routines externes
----------------------------------}
(MODULE SOFTCOMM)
EXTERNAL PROCEDURE setwopt                                              ;
EXTERNAL PROCEDURE resetwopt                                            ;
EXTERNAL PROCEDURE initrsm                                              ;
EXTERNAL PROCEDURE pinixyz                                              ;
EXTERNAL PROCEDURE piniz                                                ;
EXTERNAL PROCEDURE princage                                             ;
EXTERNAL PROCEDURE pvidange                                             ;
EXTERNAL PROCEDURE pabs         (x,y,z:INTEGER;     VAR err:INTEGER);
EXTERNAL PROCEDURE pzabs        (zposabs:INTEGER;   VAR err:INTEGER);
EXTERNAL PROCEDURE pzrel        (zposrel:INTEGER;   VAR err:INTEGER);
EXTERNAL PROCEDURE pzdip        (                   VAR err:INTEGER);
EXTERNAL PROCEDURE submerge     (npas:INTEGER;      VAR err:INTEGER);
EXTERNAL PROCEDURE defzpos      (zm,zc,zd:INTEGER;  VAR err:INTEGER);
EXTERNAL PROCEDURE defzvdg      (zpos:INTEGER ;     VAR err:INTEGER);
EXTERNAL PROCEDURE defzrcg      (zpos:INTEGER;      VAR err:INTEGER);
EXTERNAL PROCEDURE defzmvt      (zpos:INTEGER;      VAR err:INTEGER);
EXTERNAL PROCEDURE repmzpos     (VAR mzpos:INTEGER)                 ;
EXTERNAL PROCEDURE repxpos      (VAR xpos:INTEGER)                  ;
EXTERNAL PROCEDURE repypos      (VAR ypos:INTEGER)                  ;
EXTERNAL PROCEDURE repzpos      (VAR zpos:INTEGER)                  ;
EXTERNAL PROCEDURE pmpout       (npmp,npas:INTEGER; VAR err:INTEGER);
EXTERNAL PROCEDURE pmpin        (npmp,npas:INTEGER; VAR err:INTEGER);
EXTERNAL PROCEDURE refout       (npmp,npas:INTEGER; VAR err:INTEGER);
EXTERNAL PROCEDURE defvitpl     (npmp,vit:INTEGER;  VAR err:INTEGER);
EXTERNAL PROCEDURE descdstop    (npmp,npas:INTEGER; VAR err:INTEGER);
EXTERNAL PROCEDURE pmpini       (npmp:INTEGER)                      ;
EXTERNAL PROCEDURE pmptest      (npmp:INTEGER;    VAR ready:BOOLEAN);

(MODULE SCREEN)
EXTERNAL PROCEDURE CR;
EXTERNAL PROCEDURE CTEOL;
EXTERNAL PROCEDURE CTEOS;
EXTERNAL PROCEDURE CSCR;
EXTERNAL PROCEDURE RESTORE;
EXTERNAL PROCEDURE BELL         (N:INTEGER);
EXTERNAL PROCEDURE DESABLE;
EXTERNAL PROCEDURE ENABLE;
EXTERNAL PROCEDURE REVERSE;
```

```
PROCEDURE trterr (nrodev,err: INTEGER);
BEGIN
        writeln;
        reverse;
        bell(5);
        case nrodev of
        1 : WRITELN('PROBLEME A LA POMPE 1 : ERREUR #',ERR:2);
        2 : WRITELN('PROBLEME A LA POMPE 2 : ERREUR #',ERR:2);
        3 : WRITELN('PROBLEME AU BRAS      : ERREUR #',ERR:2);
        END;
        desable;
END;


PROCEDURE trtready (nropompe: INTEGER);
BEGIN
        reverse;        bell(5);
        WRITELN('POMPE ',nropompe:1,' PAS PRETE');
        desable;
END;


PROCEDURE check;
VAR     wtmp,ok : BOOLEAN;
        err     : INTEGER;
BEGIN
        wtmp := waitopt;
        pinixyz;
        resetwopt;
        pvidange;
        pmpini  (1);
        setwopt;
        pmpini  (2);
        pzabs(0,err);
        princage;
        resetwopt;
        pmpin   (1,1000,err);   IF err<>0 THEN trterr(1,err);
        setwopt;
        pmpin   (2,1000,err);   IF err<>0 THEN trterr(2,err);
        resetwopt;
        refout  (1,1000,err);   IF err<>0 THEN trterr(1,err);
        setwopt;
        refout  (2,1000,err);   IF err<>0 THEN trterr(2,err);
        pabs    (0,0,0,err);    IF err<>0 THEN trterr(3,err);
        pmptest (1,ok); IF NOT ok THEN trtready (1);
        pmptest (2,ok); IF NOT ok THEN trtready (2);
        waitopt := wtmp;
END;
```

```
PROCEDURE prackd (nrox,nroy,h,hmvt:INTEGER);
VAR     err     : INTEGER;
BEGIN
        IF (nrox<1) OR (nrox>19) OR (nroy<1) OR (nroy>12)
                THEN BEGIN nrox:=1; nroy:=1; END;
        defzmvt(hmvt,err);
        pabs((2950+131*(nrox-1)),(2300-158*(nroy-1)),h,err);
END;
```

```
PROCEDURE pxrel (xposrel: INTEGER; VAR err : INTEGER);
VAR     tmp     : INTEGER;
BEGIN
        tmp := xposcour + xposrel;
        IF (tmp<0) or (tmp>maxxpos)
        THEN err :=3
        ELSE pabs(tmp,yposcour,zposcour,err);
END;
```

```
PROCEDURE pyrel (yposrel: INTEGER; VAR err : INTEGER);
VAR     tmp     : INTEGER;
BEGIN
        tmp := yposcour + yposrel;
        IF (tmp<0) or (tmp>maxypos)
        THEN err :=3
        ELSE pabs(tmp,yposcour,zposcour,err);
END;
```

```
PROCEDURE prackqc (position,h,hmvt:integer);
const   p1x = 0;     (x-position de l'emplacement 1)
        p1y = 2360;     (y-position de l'emplacement 1)
var     rg,col,err       : INTEGER;
begin
        if (position<=0) or (position>500)
        then begin
                write('ERREUR :position rack qcc =',position);
                readln;
                exit;
             end;
        rg    := (position - 1) div 25;
        col   := (position - 1) mod 25;
        defzmvt(hmvt,err);
        pabs((p1x+(rg mod 2)*55+col*111),(p1y - rg * 118),h,err);
end;

MODEND.
```

9.5.3.4 Declarations.

```
EXTERNAL PROCEDURE check;
EXTERNAL PROCEDURE prackd (nrox,nroy,h,hmvt:INTEGER);
EXTERNAL PROCEDURE pxrel (xposrel:INTEGER;VAR err:INTEGER);
EXTERNAL PROCEDURE pyrel (yposrel:INTEGER;VAR err:INTEGER);
EXTERNAL PROCEDURE prackqc (position,h,hmvt:integer);
```

Chapter 10

Programmes de gestion des demandes.

10.1 Librairie de routines utilitaires.

```
/*****************************************************************
***************** DEFINITIONS DE GESTION D'ECRAN ****************
*****************************************************************/

#define maxint  32767
#define up      bios(4,11);
#define right   bios(4,12);
#define down    bios(4,10);
#define left    bios(4, 8);
#define cr      bios(4,13);
#define cteol   bios(4,24);
#define cteos   bios(4,17);
#define cscr    bios(4,26);
#define home    bios(4,30);
#define restore bios(4, 6);
#define bell    bios(4, 7);
#define enable  {bios(4,27);   bios(4,41);}
#define desable {bios(4,27);   bios(4,40);}
#define blink   {bios(4,27);   bios(4,52);  enable;}
#define graphic {bios(4,27);   bios(4,53);  enable;}
#define reverse {bios(4,27);   bios(4,55);  enable;}
#define lowint  {bios(4,27);   bios(4,56);  enable;}
```

```
/*****************************************************************
********* ROUTINES DE GESTION DES PORTES DE COMMUNICATON ********
*****************************************************************/

#define ctra  6
#define dataa 4
#define spda  0

/* POSITIONNE LE DTR DE LA PORTE A (1 POUR ON ,0 POUR OFF) */
int dtr(n)
int n;
{if (n == 0) {outp(ctra,5); outp(ctra,106);}
        else {outp(ctra,5); outp(ctra,234);}
}

/* SELECTIONNE LA PORTE p (1-4) DU MULTIPORTES
        RELIE A LA SORTIE A DU XEROX */
a_pseek (p)
int p;
{int i;
 for (i=0;i<=20;i++);
 outp(dataa,20); outp(dataa,(p+48));
 for (i=0;i<=20;i++);
}

/* SI UN CARACTERE EST DISPONIBLE SUR LA SORTIE A
        RENVOIE CE CARACTERE DANS C
        RENVOIE 1
 SINON  RENVOIE -1        */
int a_rec(c)
char *c;
{int i,rr0;
 outp(ctra,16);  rr0 = inp(ctra);
 if  (rr0 & 1) ( *c = inp(dataa); return 1;} else return -1;
}

/* INITIALISATION DE LA SORTIE A */
int a_ini_host()
{outp(ctra,  1);
 outp(ctra,  0);
 outp(ctra,  3);
 outp(ctra,225);
 outp(ctra,  4);
 outp(ctra, 68);
 outp(ctra,  5);
 outp(ctra,234);
 outp(spda, 14);
}
```

```
/*****************************************************************
***************** ROUTINES DE GESTION D'ECRAN ********************
*****************************************************************/

#include "a:screen.c"

/* POSITIONNE LE CURSEUR EN LIGNE L ET COLONNE C */
int pos(l,c)
int l,c;
{bios(4,27); bios(4,61); bios(4,(l+32)); bios(4,(c+32));
}

/* POSE LA QUESTION SE TROUVANT DANS ST EN POSITION (L,C)
   ET RENVOIE 1 POUR OUI, 0 POUR NON */
int oui (l,c,st,m)
char *m,*st;
int l,c;
{char x;
int i;
pos(l,c);
for (x = '?';((x != 'o')&&(x != 'O')&&(x != 'n')&&(x != 'N'));)
{switch (m){
case 'r':case 'R' : reverse; break;
case 'l':case 'L' : lowint;  break;
case 'b':case 'B' : blink;   break;
}
printf("%s (o/n) ? :",st);
desable;
bios(4,' ');
x = bios(3);
if ((x != 'o') && (x != 'O') && (x != 'n') && (x != 'N'))
        {pos(l,c); cteol; bell; bell; }
}
if ((x == 'o') uu (x == 'O')) { printf("Oui"); return (1);}
                 else         { printf("Non"); return (0);}
}
```

```
/* DESSINE UN CADRE A L'ECRAN */
cadre ()
(int  i,j;
home;cteos;
graphic;
printf("upppppppppppppppppppppppppppppppppppppppp");
printf("ppppppppppppppppppppppppppppppppppppppppppz\n");
for (i=1;i<=21;i++)
(bios(4,'U');
pos(i,78);
printf("j\n");
}
printf("5OOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOO");
printf("OOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOO:\n");
desable;
home;
pos(1,1);
}

/* DESSINE LE MEME CADRE QUE LE ROUTINE PRECEDENTE MAIS
 TIRE UNE LIGNE  HORIZONTALE SUPPLEMENTAIRE EN LIGNE 2 */
void cadrebis()
(int i;
 cadre();
 pos(2,0);
 graphic;
 bios(4,'5');
 for (i=1;i<=77;i++) bios(4,'O');
 bios(4,':');
 desable;
}


/* IMPRIME LE MESSAGE D'ERREUR ST EN LIGNE 23
             DURANT QUELQUES SECONDES */
int msg_err (st)
char  *st;
(int i;
pos(23,40);
printf("%s",st);
bell; bell;
for (i=0;i < 32000;i++);
pos (23,0);
cteol;
}
```

```
/* STOPPE LE DEROULEMENT DU PROGRAMME TANT QUE
L'UTILISATEUR NE TAPE PAS SUR UNE TOUCHE QUELCONQUE.  */
int  temporisation()
{pos(23,35);
 bell;
 printf("Tapez sur une touche pour continuer ...");
 bios(3);
 pos(23,0);
 cteol;
}


/* IMPRIME LA DATE DU CPM A L'ECRAN
 SI  N = 0  IMPRIME DATE + HEURE
     N = 1  IMPRIME DATE
     N = 2  IMPRIME HEURE        */
int scrdate (n)
int n;
{int DD,MM,YY,hh,mm,ss;
 DD = peek(call(61497,0,0,0,0)  );
 MM = peek(call(61497,0,0,0,0)+1);
 YY = peek(call(61497,0,0,0,0)+2);
 hh = peek(call(61497,0,0,0,0)+3);
 mm = peek(call(61497,0,0,0,0)+4);
 ss = peek(call(61497,0,0,0,0)+5);
 switch (n)
 {case 0 : printf("%d/%d/%d %d:%d:%d",DD,MM,YY,hh,mm,ss); break;
  case 1 : printf("%d/%d/%d",DD,MM,YY); break;
  case 2 : printf("%d:%d:%d",hh,mm,ss); break;
  default : break;
 }
}
```

```
/*************************************************************************
***************  ROUTINES DE GESTION DE L'IMPRIMANTE *************
*************************************************************************/

/*  IMPRIME LE STRING ST  */
int lptstr(st)
char *st;
{while (*st != '\0') bios(5,*(st++));
}

/* IMPRIME LE NOMBRE ENTIER N */
int lptint(n)
int n;
{int i;
 char czone[10];
 sprintf(czone,"%5.5d",n);
 for (i=0;i<=4;i++) bios(5,czone[i]);
}

/* CARIAGE RETURN - LINE FEED SUR IMPRIMANTE */
int lptnl()
{bios(5,13); bios(5,10);
}

/* IMPRIME n CARACTERES c */
int lptrepeat(n,c)
int n;
char c;
{int i;
 for (i=1;i<=n;i++) bios(5,c);
}

/* IMPRIME LES n PREMIERS CARACTERES DE st */
int lptlgr(st,n)
char *st;
int n;
{int i;
 for (i=1;i<=n;i++) bios(5,*(st++));
}

/* IMPRIME LE  CARACTERE c */
int lptcar(c)
char c;
{bios(5,c);}
```

```
/* IMPRIME LA DATE DU CPM SUR IMPRIMANTE
   SI  N = 0  IMPRIME DATE + HEURE
       N = 1  IMPRIME DATE
       N = 2  IMPRIME HEURE        */
int lptdate (n)
int n;
{int DD,MM,YY,hh,mm,ss,i;
 char zone[25];
 DD = peek(call(61497,0,0,0,0)  );  MM = peek(call(61497,0,0,0,0)+1);
 YY = peek(call(61497,0,0,0,0)+2);  hh = peek(call(61497,0,0,0,0)+3);
 mm = peek(call(61497,0,0,0,0)+4);  ss = peek(call(61497,0,0,0,0)+5);
 switch (n)
 {case 0:sprintf(zone,
        "%2.2d/%2.2d/%2.2d %2.2d:%2.2d:%2.2d",DD,MM,YY,hh,mm,ss);
           for (i=0;i<=16;i++) bios(5,zone[i]); break ;
case 1:sprintf(zone,"%2.2d/%2.2d/%2.2d",DD,MM,YY);
           for (i=0;i<=7;i++) bios(5,zone[i]); break ;
case 2:sprintf(zone,"%2.2d:%2.2d:%2.2d",hh,mm,ss);
           for (i=0;i<=7;i++) bios(5,zone[i]); break ;
  default : break;
 }
}

/* IMPRIME LE STRING ST
    EN LE SOULIGNANT   SI MODE = 's'
    EN GROS CARACTERES SI MODE = 'g'
    NORMALEMENT        SI MODE = AUTRE CHOSE
    EN CARACTERES GRAS SI gras = 'g' NORMALEMENT SINON .*/
char lpttitre(st,mode,gras)
char *st,mode,gras;
{switch (mode)
        {case 's':case 'S': lprintf("%c",29); break;
         case 'g':case 'G': lprintf("%c",30); break;
         default:;
        }
 lprintf("%s",st);
 if ((gras != 'g') && (gras != 'G')) return;
 lprintf("%c",13);
 switch (mode)
        {case 's':case 'S': lprintf("%c",29); break;
         case 'g':case 'G': lprintf("%c",30); break;
         default:;
        }
 lprintf("%s\n",st);
 lprintf("%c",31);
}
```

```
/*************************************************************************
********************** ROUTINES UTILITAIRES **********************
*************************************************************************/


/* RENVOIE 1 SI LES STRINGS S1 ET S2 SONT EGAUX SUR UNE LONGUEUR LG
   RENVOIE 0 SINON */
int egaux(s1,s2,lg)
char *s1,*s2;
int lg;
{int i;
 for (i=1;i<=lg;i++) if (*(s1++) != *(s2++)) return (0);
 return (1);
}


/* CALCULE NB A LA PUISSANCE PUISS */
int power(nb,puiss)
int nb,puiss;
{int i,tmp;
tmp = 1;
for (i=1;i <= puiss;i++)
tmp = tmp * nb ;
return(tmp);
}


/* EMET UN SON REPETE POUR ATTIRER L'ATTENTION DE L'UTILISATEUR  */
void alarm(n)
int n;
{int i,j;
 for (i=1;i<=n;i++)
 {for (j=1;j<=1500;j++);
  for (j=1;j<=20;j++) bell;
 }
}
```

```
/*****************************************************************
*********  ROUTINES DE GESTON DES TABLEAUX DE DEMANDES  **********
*****************************************************************/

/* RENVOIE TRUE SI L'ANALYSE NRO "analyse" EST DEMANDEE
    DANS LE TABLEAU DE DEMANDES "dde"   */

int testdde (dde,analyse)
int *dde,analyse;
{if (dde[analyse/16] & power(2,(analyse%16)))
        return (1); else return (0);
}



/* POSITIONNE A OUI LA DEMANDE DE L'ANALYSE NRO "analyse"
    DU TABLEAU DE DEMANDES "dde"   */

int setdde (dde,analyse)
int *dde,analyse;
{dde[analyse/16] = (dde[analyse/16] u power(2,analyse%16));
}



/* POSITIONNE A NON LA DEMANDE DE L'ANALYSE NRO "analyse"
    DU TABLEAU DE DEMANDES "dde"   */

int resetdde (dde,analyse)
int *dde,analyse;
{dde[analyse/16] = (dde[analyse/16] & (~(power(2,analyse%16))));
}
```

10.2 Routine HELP

10.2.1 Introduction.

Cette routine est actuellement hors service dans le programme d'encodage; elle y figurait au début de son utilisation mais maintenant que les utilisateurs sont bien familiarisés avec les différentes possibilités du programme elle n'est plus nécessaire et elle à laissé de la place pour d'autres fonctions.

10.2.2 Texte source.

```
/* ROUTINE  HELP POUR LE PROGRAMME D'ENCODAGE DES DEMANDES */
int help(n)
int n;
{switch (n){
case 1 :
 cadre();
 pos(2,18); printf("UTILISEZ UNE DES TOUCHES SUIVANTES :");
 pos(3,15); printf("----------------------------------------");
 pos(6,8);
 printf("<fleche vers le haut>    pour remonter  au chiffre precedent");
 pos(7,8);
 printf("<fleche vers le bas >    pour descendre au chiffre suivant");
 pos(8,8);
 printf("<point>                    idem");
 pos(9,8);
 printf("<chiffre de 0 -> 9>      pour taper un numero d'analyse");
 pos(10,8);
 printf("<esc>                    pour passer a une autre reference");
 pos(11,8);
 printf("<del>                    pour effacer le chiffre en cours");
 pos(12,8);
 printf("<?>                      pour imprimer ce texte");
 temporisation();
 break;
case 2 :
 cadre();
 pos(2,18); printf("UTILISEZ UNE DES TOUCHES SUIVANTES :");
 pos(3,15); printf("----------------------------------------");
 pos(6,8);
 printf("<fleche vers le bas >    pour commencer a encoder les demandes"
 pos(7,8);
 printf("<lettre de A -> Z>       pour selectionner la reference");
 pos(8,8);
 printf("<fleche vers la droite>  pour taper un numero particulier");
 pos(9,8);
 printf("<?>                      pour imprimer ce texte");
 pos(10,8);
 printf("<esc>                    pour arreter le programme");
 temporisation();
 break;
        }
}
```

10.3 Le programme d'encodage.

10.3.1 Signification des principales variables.

t900 a t915 — tableaux qui contiennent les numéros d'analyse repris sous le code 900 ... 915

d900 a d915 — dimension de ces tableaux

nomencla — tableau contenant les noms des différentes analyses existantes

out_ref_buffer et in_ref_buffer — ces zones peuvent contenir 16 références chacune

dde — tableau de 64 entiers soit 1024 bits pour y mémoriser les demandes

lst_ana — tableau de 195 entiers où sont rangés les numéros des analyses introduites à l'écran avant de les mémoriser sour forme de bits dans dde

tab_val_cour — contient la dernière référence utilisée pour chaque lettre

10.3.2 Description des fonctions.

REMARQUE:

1. Avant de commencer cette description, je voudrais définir une nouvelle notion: l'écran, pendant l'encodage affiche la référence en cours ainsi que les numéros d'analyse introduits. On peut introduire 195 numéros et chaque numéro va se mettre à un endroit déterminé sur l'écran. Il y a donc 195 cellules pouvant contenir un chiffre et chacune d'entre elles a une coordonnée en ligne et en colonne.

2. Ce programme, bien qu'il offre diverses fonctions à l'utilisateur, n'a pas de menu au sens ou on l'entend d'habitude. Pour faciliter le compréhension je parlerai de niveau supérieur (= menu principal) et de niveau inférieur (= menu secondaire). Les choix du niveau supérieur peuvent s'effectuer lorsque le curseur se trouve à gauche de la référence, le niveau inférieur correspond aux opérations

que l'on peut faire lorsqu'on est en train d'introduire les numéros des analyses demandées. Ces différents choix à l'intérieur de chaque niveau seront détaillées plus loin.

3. Lorsque je dis on "encode une référence", ou on "modifie une référence" , "référence" doit être pris dans le sens "patient" et pas numéro de référence; autrement dit, ce n'est pas le numéro de référence que l'on modifie ou que l'on encode mais bien toutes les demandes relatives à cette référence.

int ligne(n) renvoie la ligne de la cellule n (1<=n<=195)

int colonne(n) renvoie la colonne de la cellule n (1<=n<=195)

int blanc(n) efface la cellule n

int map_vide() efface l'écran, y dessine un cadre et y affiche des commentaires

int chger_val() procédure interactive qui permet de modifier une des valeurs du tableau tab_val_cour

int tab_write(tableau)
affiche les valeurs >=0 de tableau dans les cellules de l'écran

int posit(n) positionne le curseur devant la cellule n

int concat_ref(reference)
renvoie dans référence, sous forme de string la référence en cours de traitement

int get_tab(tab_dde)
effectue toute la gestion d'écran pour l'introduction des numéros d'analyse. Ce module gère donc le niveau inférieur. Les opérations possibles sont:

- reculer à la cellule précédente
- avancer à la cellule suivante
- annuler tout ce qui a été encodé sur l'écran en cours et remonter au niveau supérieur
- effacer la cellule où se trouve le curseur
- signaler qu'on a terminé d'introduire les numéros d'analyse et enregistrer

- demander du secours (HELP)

renvoie 0 s'il faut traiter le tableau tab_dde, 1 sinon (on a annulé)

int load_nomencl()
charge dans la table nomencla la liste des noms d'analyse qui se trouvent dans le fichier FANAL.DAT

int ref_write() affiche dans le haut de l'écran la référence en cours

int sortie() stoppe le programme après avoir fermé les fichiers

int search_ref(s_ref)
renvoie la position de la référence s_ref dans le fichier FILREF ou -1 si elle ne s'y trouve pas.

int enreg(ana) crée une ligne du type (ana,référence en cours) dans le fichier TRVLST

int select_ref ()
lit à l'écran la référence tapée par l'utilisateur et si elle existe dans le fichier FILREF, renvoie 1 et remplit lst_ana avec les analyses demandées pour cette référence

int trt_ref() gère le niveau supérieur et permet les opérations suivantes:

- changer de lettre de référence
- signaler que l'on veut commencer a introduire les numéros des analyses demandées pour la référence affichée
- signaler que l'on veut sélectionner une référence particulière
- signaler que l'on veut changer la valeur de la dernière référence utilisée pour une lettre donnée
- stopper le programme

int testnro(n,ipos,liste_ana)
si n est un numéro de code (900 a 915), affiche dans les cellules de l'écran toutes les analyses reprises sous ce code et remplit le tableau

liste_ana avec ces mêmes valeurs

int imp_ana(tab,dim,ipos,liste_ana)
affiche à l'écran à partir de la cellule ipos, les dim numéros d'analyse contenus dans tab et les range dans liste_ana

10.3.3 Programme principal.

- initialisation des tables des valeurs codes avec les numéros des analyses reprises sous chaque code
- chargement des noms d'analyse
- ouverture du fichier des références FILREF;
  * si le fichier existe on rajoutera a la fin
  * sinon, on crée un nouveau fichier
- ouverture du fichier des demandes FILDDE et même traitement que pour FILREF
- chargement des valeurs de la dernière référence utilisée dans chaque lettre
- chargement des numéros des analyses pour lesquelles il ne faut pas créer de petites listes (ce sont les analyses déjà reprises sur les tableaux)
- traitement répété:
  * traitement niveau supérieur (Cfr trt_ref)
  * s'il y a une référence encodée ou modifiée, on l'enregistre ainsi que ses demandes et on écrit dans le fichier TRVLST les lignes qui serviront à établir les petites listes de travail. Pendant ce temps, on imprime la référence et ses demandes (Cfr listing lan12 appendice L du premier volume)
  * si c'est une nouvelle référence, (en opposition a une référence que l'on a modifiée donc qui se trouvait déjà dans le fichier) on incrémente de 1 cette référence

10.3.4 Utilisation.

- L'écran 1 demande si on désire un listing reproduisant ce que l'on a encodé à l'écran

- écran 2 : le programme trouve les fichiers vides et demande si on commence bien une nouvelle journée

- écran 3 : on peut commencer à encoder; au départ, le programme affiche la prochaine référence commencant par "J". Si on veut encoder une référence commencant par "Z" on presse simplement sur la touche "Z" et on obtient l'écran 4.

- Si on tape sur "+", le programme nous permet de changer le Z66667 en Z66661. (ecran 5)

- Si on avait tapé sur la flèche vers la droite, on aurait obtenu l'écran 6 et on aurait pu demander pour consulter/modifier une référence particulière.

- Reprenons à l'écran 7: le curseur se trouve devant REFERENCE; si on veut commencer à encoder les analyses demandées pour cette référence, on tape sur la flèche vers le bas et le curseur se trouve devant la première cellule de l'écran. (ecran 8)

- On peut alors taper les différents numéros d'analyses demandées. La flèche vers le bas permet de passer à la cellule suivante, celle vers le haut, à la cellule précédente. La touche "del" permet d'effacer la cellule se trouvant à droite du curseur.

- L'écran 9 montre le résultat de l'introduction d'une série de numéros d'analyses

- A l'écran 10, on a supprimé les analyses 4, 5 et 6 et on a ajouté les analyses 51, 52 et 53.

- Une fois terminé, on tape sur la touche "esc" pour enregistrer le contenu de l'écran. Lorsque cet enregistrement est terminé, le programme affiche l'écran 11 où la référence a été incrementée de 1.

- Ecran 12 : on a introduit les demandes pour la référence Z66662.

- Ecran 13 : le programme attend que l'on introduise les demandes pour la référence Z66663

- on voudrait modifier la Z66661 et on tape sur la flèche vers la droite pour obtenir l'ecran 14.

- On introduit Z66661 et le programme va rechercher les demandes pour cette référence. (écran 15)

- Lorsqu'il a trouvé, il affiche ces demandes (écran 16) et on peut de nouveau circuler parmi celles-ci et effectuer les modifications. (écran 17)

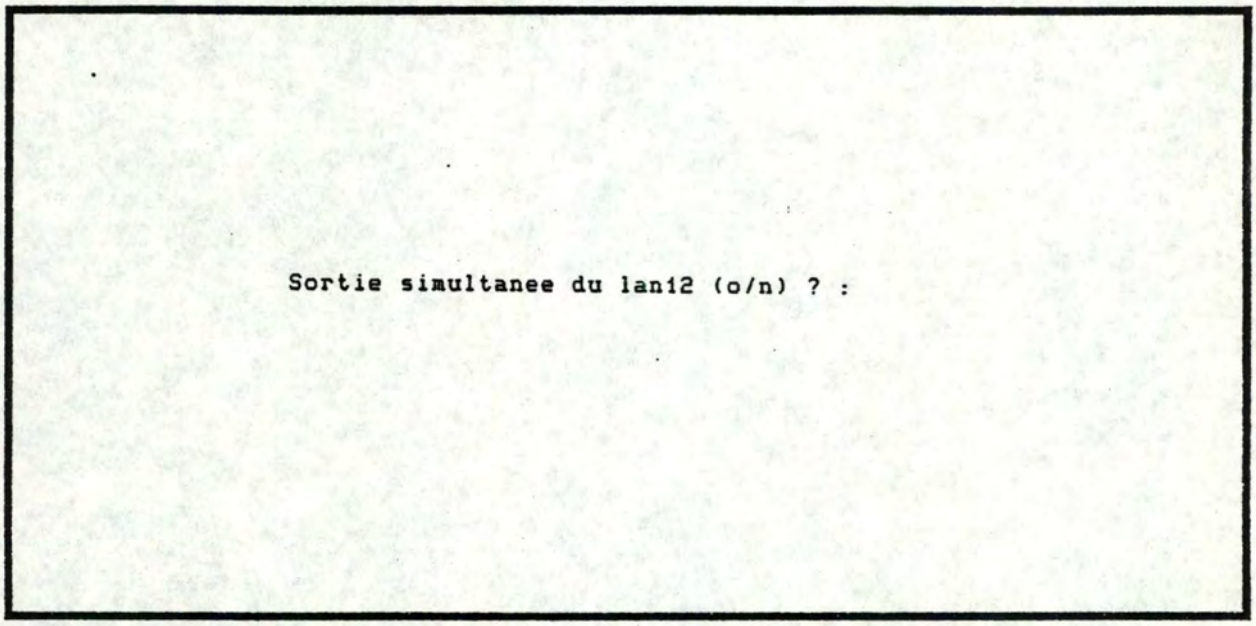
- Lorsque les corrections sont terminées, on tape sur "esc" pour enregistrer et il demande s'il doit tenir compte de ces corrections dans les petites listes de travail. (écran 18)

- Il réaffiche ensuite l'écran normal pour encoder la référence suivante (ecran 19)

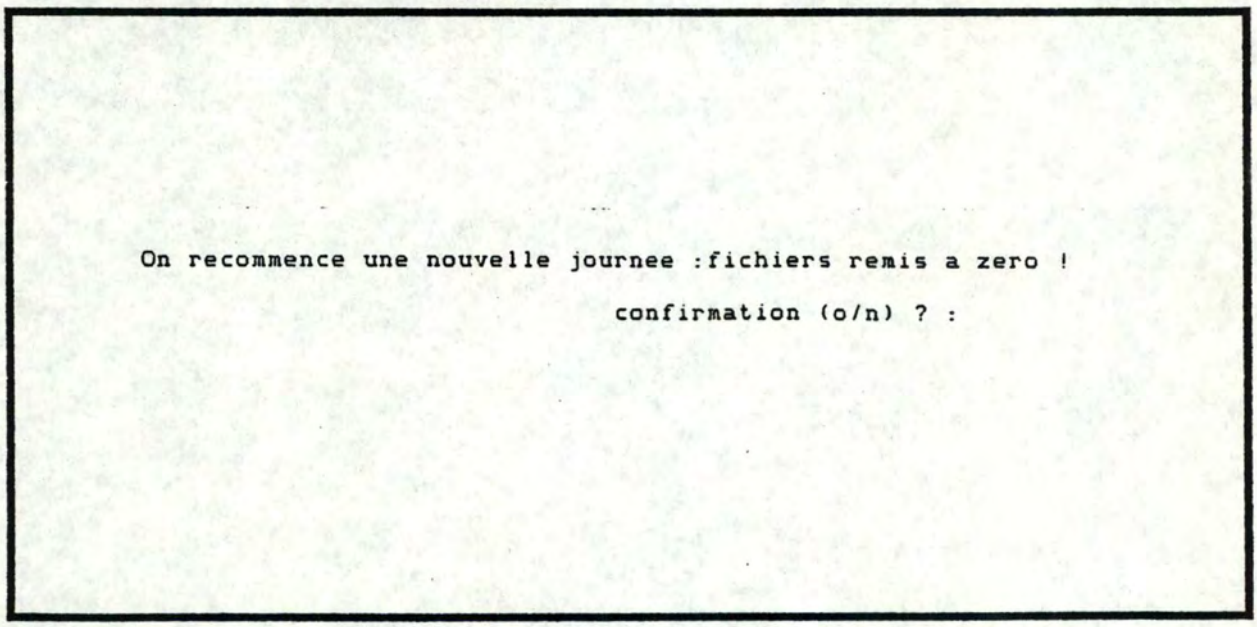
- Si on veut encoder une référence commençant par N on tape sur "N" et on obtient l'écran 20.

- Si on veut sortir du programme, on tape sur "esc" et il demande confirmation. (écran 21)

écran 1



écran 2



écran 3

```
21/4/84 15:15:0        ref:  1        ENCODAGE DES DEMANDES D'ANALYSES

                      REFERENCE  J50020
```

écran 4

```
21/4/84 15:15:0        ref:  1        ENCODAGE DES DEMANDES D'ANALYSES

                      REFERENCE  Z66667
```

écran 5

```
La prochaine reference commencant par 'Z' est Z66667 .
VOULEZ-VOUS CHANGER CETTE VALEUR (o/n) ? : Oui

Veuillez donner la nouvelle valeur desiree : Z66661

ANCIENNE VALEUR : Z66667 , NOUVELLE VALEUR : Z66661

CONFIRMEZ-VOUS CE CHANGEMENT (o/n) ? :
```

écran 6

```
21/4/84 15:18:12        ref:  1        ENCODAGE DES DEMANDES D'ANALYSES

                          REFERENCE  Z_____
```

écran 7

21/4/84 15:18:12	ref: 1	ENCODAGE DES DEMANDES D'ANALYSES
	REFERENCE Z66661	

écran 8

21/4/84 15:18:12	ref: 1	ENCODAGE DES DEMANDES D'ANALYSES
	REFERENCE Z66661	

écran 9

21/4/84 15:18:12 ref: 1 ENCODAGE DES DEMANDES D'ANALYSES

REFERENCE Z66661

0	1	2	3	4	5	6	7	25	22	39	123	124
125	126	127	128	251	541	542	543	555	556	55	56	57
58	59	60	61	62	63	64	65	66	67	457	458	568
19	33											

écran 10

21/4/84 15:18:12 ref: 1 ENCODAGE DES DEMANDES D'ANALYSES

REFERENCE Z66661

0	1	2	3				7	25	22	39	123	124
125	126	127	128	251	541	542	543	555	556	55	56	57
58	59	60	61	62	63	64	65	66	67	457	458	568
19	33						51	52	53			

écran 11

```
21/4/84 15:26:56        ref:  2        ENCODAGE DES DEMANDES D'ANALYSES

                        REFERENCE  Z66662
```

écran 12

```
21/4/84 15:28:2         ref:  3        ENCODAGE DES DEMANDES D'ANALYSES

                        REFERENCE  Z66662

0     2     3     5     10    99
```

écran 13

21/4/84 15:29:7	ref: 3	ENCODAGE DES DEMANDES D'ANALYSES
REFERENCE Z66663		

écran 14

21/4/84 15:29:7	ref: 3	ENCODAGE DES DEMANDES D'ANALYSES
REFERENCE Z_____		

écran 15

```
21/4/84 15:30:41        ref:  3        ENCODAGE DES DEMANDES D'ANALYSES

                          REFERENCE  Z66661      En cours ...
```

écran 16

```
21/4/84 15:30:41        ref:  3        ENCODAGE DES DEMANDES D'ANALYSES

                          REFERENCE  Z66661

0     1     2     3     7     19    22    25    33    39    51    52    53
55    56    57    58    59    60    61    62    63    64    65    66    67
123   124   125   126   127   128   251   457   458   541   542   543   555
556   568
```

écran 17

21/4/84 15:30:41 ref: 3 ENCODAGE DES DEMANDES D'ANALYSES

REFERENCE Z66661

0	1	2	3	7	19	22	25	33	39	51	52	53
55	56	57	58	59	60				64	65	66	67
123	124	125	126	127	128	251	457	458	541	542	543	555
556	568	333	334									

écran 18

21/4/84 15:30:41 ref: 3 ENCODAGE DES DEMANDES D'ANALYSES

REFERENCE Z66661

0	1	2	3	7	19	22	25	33	39	51	52	53
55	56	57	58	59	60				64	65	66	67
123	124	125	126	127	128	251	457	458	541	542	543	555
556	568	333	334									

generation des listes de travail (o/n) ? :

écran 19

21/4/84 15:35:15	ref: 3	ENCODAGE DES DEMANDES D'ANALYSES
	REFERENCE Z66663	

écran 20

21/4/84 15:35:15	ref: 3	ENCODAGE DES DEMANDES D'ANALYSES
	REFERENCE N22238	

écran 21

21/4/84 15:35:15	ref: 3	ENCODAGE DES DEMANDES D'ANALYSES
	REFERENCE N22238	

confirmez la sortie du programme (o/n) ? :

V52192

0	S.VITESSE DE SEDIMENTAT.
1	S.HEMOGLOBINE
2	S.GLOBULES ROUGES
3	S.GLOBULES BLANCS
4	S.FORMULE LEUCOCYTAIRE
5	S.HEMATOCRITE
54	S.GROUPES SANGUINS ABO
55	S.FACTEUR RH
66	S.TEMPS DE SAIGNEMENT
71	S.CEPHALINE (TEMPS DE)
74	S.PLAQUETTES NUMERATION
245	S.STREPTOZYME
255	S.ANTI-PROTEINE M
258	S.ANTI-HYALURONIDASE
314	S.PROTHROMBINE ENFANT
315	S.THROMBOTEST ENFANT
46	S.ASL (STREPTO)

J59495

96	S.SODIUM
97	S.POTASSIUM
102	S.CALCIUM
105	S.MAGNESIUM SERIQUE
327	S.MAGNESIUM ERYTHROCYTAI
35	S.FER

J59549

0	S.VITESSE DE SEDIMENTAT.
1	S.HEMOGLOBINE
2	S.GLOBULES ROUGES
3	S.GLOBULES BLANCS
4	S.FORMULE LEUCOCYTAIRE
5	S.HEMATOCRITE
6	S.UREE
7	S.GLUCOSE
8	S.CHOLESTEROL TOTAL
10	S.LIPIDES TOTAUX
11	S.LIPIDOGRAMME
18	S.PROTEINES TOTALES
19	S.PROTEINES ELECTROPHOR.
20	S.THYMOL
23	S.TRIGLYCERIDES
24	S.KUNKEL ZINC
26	S.SGOT

10.3.5 Texte source.

```
#include "A:BDSCIO.H"
#include "a:screen.c";
/*#include "a:help.c";*/

/*  DEFINITION DES CONSTANTES */
#define max_ana  195
#define fl_haut    1
#define fl_bas     2
#define fl_droite  3
#define fl_gauche  4
#define ctrl_g     7
#define escape    27
#define point     46
#define cr        13
#define interrog  63
#define del      127
#define lf        10
#define moins     45
#define plus      43
#define next_page 12
#define d900  6
#define d901  7
#define d902  4
#define d903  5
#define d904  8
#define d905  12
#define d906  9
#define d907  5
#define d908  4
#define d909  4
#define d910  4
#define d911  4
#define d912  6
#define d913  10
#define d914  7
#define d915  20
```

```
/* TYPES ET VARIABLES GLOBALES */
int t900[d900];
int t901[d901];
int t902[d902];
int t903[d903];
int t904[d904];
int t905[d905];
int t906[d906];
int t907[d907];
int t908[d908];
int t909[d909];
int t910[d910];
int t911[d911];
int t912[d912];
int t913[d913];
int t914[d914];
int t915[d915];

char iobuf[BUFSIZ];
char trvlst[BUFSIZ];
char trvbck[BUFSIZ];

struct  { char lib[26]; } nomencla[1024];

int taille_nomencla;

struct nro_ref {char nro[6];};

char swlst[1024],tmps[13];

struct st_buf   {char lettre;
                 char ref[5];
                 char car_return  ;
                 char line_feed   ;
                }in_ref_buffer[16],out_ref_buffer[16],
                 ref_buffer[32],*ptr_buf;

int filref,fildde,filval,lst_ana[max_ana+1],dde[64],genlist;
struct nro_ref  tab_val_cour[26];
char lettre_cour , *ptr_char,ref_select[7];
int pos_ref,nb_ref,nro_sect_ref,ref_existante,nn,refnb,imp_lan12;
```

```
/* CHARGE LES VALEURS DU FICHIER FILVAL DANS LA TABLE tab_val_cour */
int load_val_ref ( )
{ int i,j;
  setmem(ref_buffer,256,'\0');
  read(filval,ref_buffer,2);
  for (i=0;i<=25;i++)
          {for (j=0;j<=4;j++)
           tab_val_cour[i].nro[j] = ref_buffer[i].ref[j];
           tab_val_cour[i].nro[5] = '\0';
          }
}

/* SAUVE tab_val_cour DANS LE FICHIER FILVAL */
int save_val_ref ()
{ int i,j;
  for (i=0;i<=25;i++)
        {  for (j=0;j<=4;j++)
                 ref_buffer[i].ref[j] = tab_val_cour[i].nro[j] ;
           ref_buffer[i].lettre = 65 + i ;
           ref_buffer[i].car_return = 13;
           ref_buffer[i].line_feed = 10;
        }
  seek (filval,0,0);
  write(filval,ref_buffer,2);
}
/* INREMENTE DE 1 LA REFERENCE reference  */
int plus1(reference)
struct nro_ref  *reference;
{int i;
 char *ptr;
 ptr = reference ;
 ptr++;
 ptr++;
 ptr++;
 ptr++;
 (*ptr)++;
 for (i=1;i<=4;i++)
        if ((*ptr) > '9') {*ptr = '0';
                           (*(--ptr))++;
                          }
 if ((ptr == reference) && (*ptr > '9'))
        {*(ptr++) = '1';
         *(ptr++) = '0';
         *(ptr++) = '0';
         *(ptr++) = '0';
         *ptr     = '1';
        }
}
```

```
/* RENVOIE LE NUMERO DE LIGNE POUR LE NUMERO D'ANALYSE N */
int ligne (n)
int n;
{ return (((n-1)/13)+7); }


/* RENVOIE LE NUMERO DE COLONNE POUR LE NUMERO D'ANALYSE N */
int colonne (n)
int n;
{ return (((n-1)%13)*6+2); }



/* IMPRIME DES BLANCS A LA POSITION DU NUMERO D'ANALYSE N */
int blanc (n)
int n;
{pos(ligne(n),colonne(n));
 bios(4,' '); bios(4,' '); bios(4,' '); bios(4,' ');
}


/* INITIALISE L'ECRAN AVEC CADRE ET COMMENTAIRES.  */
int map_vide ()
{int i;
 cadre();
 pos(5,0);
 graphic;
 bios(4,'5');
 for (i = 1;i <= 77; i++) bios(4,'0');
 bios(4,':');
 pos(2,0);
 bios(4,'5');
 for (i = 1;i <= 77; i++) bios(4,'0');
 bios(4,':');
 desable;
 pos(1,2);
 scrdate(0);
 pos(1,25);
 printf("ref:%3.3d",refnb);
 pos(1,40);
 printf("ENCODAGE DES DEMANDES D'ANALYSES");
 pos(4,31);
 printf("REFERENCE  ______");
}
```

```
/* PERMET DE MODIFIER UNE VALEUR DE FILVAL.DAT */
int chger_val ()
{char rr[7],rn[7],c;
 int i;
 home; cteos;
 concat_ref(rr);
 rr[6] = '\0';
 printf("\n\n\n\n\n\nLa prochaine reference ");
 printf("commencant par '%c' est %s .",lettre_cour,rr);
 if (oui(7,0,"VOULEZ-VOUS CHANGER CETTE VALEUR",'r'))
  {pos(10,0); printf(
   "Veuillez donner la nouvelle valeur desiree : %c_____",lettre_cour);
   pos(10,46);
   for (i=1;i<=5;) {c = bios(3);
                 if (! isdigit(c)) {msg_err("Des chiffres uniquement")
                                    pos(10,45+i);}
                              else  {bios(4,c);
                                    rn[i++] = c;}
                }
   rn[0] = lettre_cour;
   rn[6] = '\0';
   pos(13,0);
   printf("ANCIENNE VALEUR : %s , NOUVELLE VALEUR : %s ",rr,rn);
   if (oui(15,0,"CONFIRMEZ-VOUS CE CHANGEMENT",'r'))
    {for (i=1;i<=5;i++)
           tab_val_cour[lettre_cour - 65].nro[i-1] = rn[i];
     save_val_ref();
     reverse;
     printf("\n ATTENTION ! Si vous avez recule un compteur, n'oubliez"
     printf(" pas de retirer la\n                    mauvaise carte B lors
;
     printf("du lan12 intermediaire et du final.\n              Sur les"
     printf(" listes et les tableaux, les demandes de toutes les\n");
     printf("                  references identiques sont cumulees ... ")
     printf("Veillez a biffer les\n                  mauvaises.");
     desable;
     temporisation();
    }
  }
   map_vide();
   ref_write();
}
```

```
/* AFFICHE A L'ECRAN TOUS LES NOMBRES DE TABLEAU >= 0 */
int tab_write (tableau)
int *tableau;
{int i,max,*p;
 max = ((*tableau == -1)?max_ana:*tableau);
 p = tableau;
 p++;
 for (i = 1;i <= max;i++)
 {if (*p != -1)
         {pos(ligne(i),colonne(i));
          printf("%d",*p);
         }
          p++;
 }
}


/* POSITIONNE LE CURSEUR DEVANT LA POSITION DU NOMBRE N */
int posit (n)
int n;
{pos(ligne(n),colonne(n)-1);
}



/* RENVOIE DANS "RF", LA REFERENCE EN COURS  (CARACTERE + NUMERO)  */
int concat_ref (rf)
char *rf;
{int i;
if (ref_existante) for (i=0;i<=5;i++) *(rf++) = ref_select[i];
        else       {rf[0] = lettre_cour;
                    for (i=1;i<=5;i++)
                       rf[i] = tab_val_cour[lettre_cour - 65].nro[i-1];
                   }
}
```

```
/* GERE L'ENCODAGE DES NUMEROS D'ANALYSE EN RENVOIE CES NUMEROS
   DANS TAB_DDE  */
int get_tab (tab_dde)
int *tab_dde;
{int nb_en_cours,nb,icour,maxicour,i;
 char c;
 maxicour = max_ana;
 nb_en_cours = nb = 0;
 icour = 1;
 posit(icour);
 for (c = 0;c != escape;)
 {c = bios(3);
  if (isdigit(c)) {if (tab_dde[icour] != -1) {tab_dde[icour] = -1;
                                              blanc(icour);
                                              posit(icour);}
                   if (! nb_en_cours)  {nb_en_cours = 1;
                                        nb = 0;
                                        bios(4,' '); }
                   nb = nb*10 + c - 48 ;
                   bios (4,c);
                  }
           else  switch (c)
                 { case fl_haut :
                      if (nb_en_cours) { tab_dde[icour] = nb;
                                         testnro(nb,&icour,tab_dde);
                                         nb_en_cours = 0;}
                            icour = (icour==1?maxicour:(--icour));
                            posit(icour);
                            break;

                   case fl_bas : case point : case cr :
                      if (nb_en_cours) { tab_dde[icour] = nb;
                                         testnro(nb,&icour,tab_dde);
                                         nb_en_cours = 0; }
                      icour = (icour==maxicour?1:(++icour));
                      posit(icour);
                      break;

                   case moins :
                      return -1;
                      break;

                   case del :
                      nb_en_cours = 0;
                      tab_dde[icour] = -1;
                      blanc(icour);
                      posit(icour);
                      break;
```

```
                    case ctrl_g :
                        if (nb_en_cours) { tab_dde[icour] = nb;
                                           testnro(nb,&icour,tab_dde);
                                           nb_en_cours = 0;}
                        icour = (icour==maxicour?1:(++icour));
                        map_vide();
                        ref_write( );
                        tab_write(tab_dde);
                        posit(icour);
                        break;

                     case lf :
                        for (i=0;i<=5;i++) lptnl();
                        break;

                     case escape :
                        if (nb_en_cours) { tab_dde[icour] = nb;
                                           testnro(nb,&icour,tab_dde);
                                           nb_en_cours = 0;}
                        break;
/*                   case interrog :
                        if (nb_en_cours) { tab_dde[icour] = nb;
                                           testnro(nb,&icour,tab_dde);
                                           nb_en_cours = 0;
                                         }
                        icour = (icour==maxicour?1:(++icour));
                        help(1);
                        map_vide();
                        ref_write( );
                        tab_write(tab_dde);
                        posit(icour);
                        break;
*/
                     default :
                        if (nb_en_cours) {nb_en_cours = 0;
                                          blanc(icour);}
                        msg_err("Caractere non valable !!");
                        posit(icour);
                        break;

                }
 }
 return 0;
}
```

```
/* CHARGE DANS "NOMENCLA" LA LISTE DES NOMS D'ANALYSES */
load_nomencl()
(int i,n,x;
 pos(15,10);
 blink; printf("En cours ...");
 desable;
 char tmp[26];
 fopen ("a:fanal.dat",iobuf);
 for (n=0 ;fscanf(iobuf,"%d %s\n",&x,tmp)>0;n++ )
        (for (;x != n;)
                strcpy(nomencla[n++].lib,"**************************");
         strcpy(nomencla[n],tmp);
        }
        taille_nomencla = --n;
 fclose(iobuf);
}


/* AFFICHE A L'ECRAN LA REFERENCE EN COURS */
int ref_write( )

(pos(4,42);
 if (ref_existante) printf("%s",ref_select);
         else    (  bios(4,lettre_cour);
                    printf("%s",tab_val_cour[lettre_cour - 65]);
                    pos(4,41);
                 }
}


/* EFFECTUE LA CLOTURE ET LA SORTIE DU PROGRAMME */
int sortie ()
(if (oui(23,10,"confirmez la sortie du programme",'n') )
        (home;
         cteos;
         close(filref);
         close(fildde);
         close(filval);
         putc(CPMEOF,trvlst);
         fclose(trvlst);
         if (imp_lan12) lptcar(next_page);    /* SAUT DE PAGE */
         exit();
        }
 pos(23,0);
 cteol;
 pos(4,41);
}
```

```
/* RENVOIE LA POSITION DE LA REFERENCE s_ref
  DANS LE FICHIER FILREF (0 a n)
         ou  -1 SI ELLE NE S'Y TROUVE PAS      */
int search_ref (s_ref)
char *s_ref;
{int i,fd_ref_sv,fd_dde_sv,nro_ref,nro_sect;
 fd_ref_sv = tell(filref);
 fd_dde_sv = tell(fildde);
 nro_ref = -1;
 if (seek(filref,0,0) == -1) return (-1);
 for (nro_sect = 0 ;(read(filref,in_ref_buffer,1) > 0);nro_sect++ )
    {for (i=0;i<=15;i++)
        if (egaux(&in_ref_buffer[i].lettre,s_ref,6) )
                {nro_ref = i;
                 break;
                }
         if (nro_ref != -1)  break;
    }
 if (nro_ref != -1) return((nro_sect * 16) + nro_ref);
            else    return (-1);
 seek(filref,fd_ref_sv,0);
 seek(fildde,fd_dde_sv,0);
}
```

```
/* ENREGISTRE POUR L'ANALYSE ANA UNE LIGNE DE LISTES DE TRAVAIL
   DANS TRVLST */
int enreg(ana)
int ana;
{char rr[7];
 if (swlst[ana] == 'n')  return;
 concat_ref(rr);
 rr[6] = '\0';
 fprintf(trvlst,"%4.4d%s\n",ana,rr);
}


/* SAUVETAGE DES FICHIERS EN CAS DE PANNE DE COURANT */
int gen_save()
{int sv1,sv2;
bell; bell; bell; bell; bell; bell; bell;
pos(23,10);
printf("Sauvetage .....    ");
sv1 = tell(fildde);
sv2 = tell(filref);
close (fildde);
close (filref);
fildde = open("b:fildde.dat",2);
filref = open("b:filref.dat",2);
seek (fildde,sv1,0);
seek (filref,sv2,0);
putc(CPMEOF,trvlst);
fclose(trvlst);
unlink ("b:trvlst.bck");
rename("b:trvlst.dat","b:trvlst.bck");
fopen("b:trvlst.bck",trvbck);
fcreat("b:trvlst.dat",trvlst);
while(fgets(tmps,trvbck) != 0) fputs(tmps,trvlst);
fclose(trvbck);
}
```

```
/* LIT A L'ECRAN UNE REFERENCE ;
   SI ELLE EXISTE DANS LE FICHIER
   - REMPLIT "LST_ANA" AVEC LES DEMANDES CORRESPONDANT A CETTE REFERENC
   - POSITIONNE REF_EXISTANTE A 1
   - MET CETTE REFERENCE DANS "REF_SELECT"
   - RENVOIE 1
   SINON RENVOIE 0     */
int select_ref( )
{int i,j,rescomp;
 char *pt,c;
 pos(4,43);  printf("_____");
 pos(4,43);
 for (i=1;i<=5;) {c = bios(3);
                  if (! isdigit(c)) {msg_err("Des chiffres uniquement")
                                     pos(4,42+i);}
                                else    {bios(4,c);
                                     ref_select[i++] = c;}
                 }
 ref_select[0] = lettre_cour;
 ref_select[6] = '\0';
 pos(4,41);
 pt = ref_select;
 pt++;
 rescomp = strcmp(tab_val_cour[lettre_cour - 65],pt);
  if (rescomp <= 0)
        {msg_err("Pas encore encodee !!");
         return (0);}
  else {pos(4,54); printf("En cours ...");
        pos_ref = search_ref(ref_select);
        if (pos_ref >= 0)
                {seek(fildde,pos_ref,0);
                if (read(fildde,dde,1) <= 0) return (0);
                for (i=0;i <= (max_ana) ; i++)  lst_ana[i] = -1;
                j = 1;
                for (i=0;i <= 1023 ; i++)
                   if (testdde(dde,i)) lst_ana[j++] = i;
                lst_ana[0] = --j;
                ref_existante = 1;
                pos(4,54); printf("            ");
                return(1);
                }
        else    {msg_err ("Pas dans le fichier");
                pos(4,54);
                printf("            ");
                return (0);}
        }
}
```

```
/* GESTION DE LA REFERENCE : RENVOIE 0 TANT QU'UNE BONNE REFERENCE N'A
   PAS ETE SELECTIONNEE     */
int trt_ref( )
{char c;
 int trouve,i;
 ref_existante = 0;
 ref_write( );
 pos(4,41);
 for (c='?';(c != fl_bas); )
 {c = toupper(bios(3));
  if (isupper(c)) {lettre_cour = c;
                   ref_write( );
                  }
        else switch (c)
               {case fl_bas: case point :
                        return(1);
                        break;

                case fl_droite :
                        trouve = select_ref( );
                        return(trouve);
                        break;

                case lf :
                        for (i=0;i<=5;i++) lptnl();
                        break;

                case escape :
                        sortie ();
                        break;

/*              case interrog :
                        help(2);
                        map_vide();
                        ref_write( );
                        pos(4,41);
                        break;
*/
                case plus:
                        chger_val();
                        break;

                default :
                        msg_err("Caractere non valable !!");
                        pos(4,41);
                        break;
                }
 }
}
```

```
/* VERIFIE SI n EST UN NUMERO DE CODE ET AGIT EN CONSEQUENCE */
int testnro(n,ipos,liste_ana)
int n,*ipos,*liste_ana;
{if ((n < 900) uu (n >1000)) return;
 else switch(n)
 {case 900: imp_ana(t900,d900,ipos,liste_ana); break;
  case 901: imp_ana(t901,d901,ipos,liste_ana); break;
  case 902: imp_ana(t902,d902,ipos,liste_ana); break;
  case 903: imp_ana(t903,d903,ipos,liste_ana); break;
  case 904: imp_ana(t904,d904,ipos,liste_ana); break;
  case 905: imp_ana(t905,d905,ipos,liste_ana); break;
  case 906: imp_ana(t906,d906,ipos,liste_ana); break;
  case 907: imp_ana(t907,d907,ipos,liste_ana); break;
  case 908: imp_ana(t908,d908,ipos,liste_ana); break;
  case 909: imp_ana(t909,d909,ipos,liste_ana); break;
  case 910: imp_ana(t910,d910,ipos,liste_ana); break;
  case 911: imp_ana(t911,d911,ipos,liste_ana); break;
  case 912: imp_ana(t912,d912,ipos,liste_ana); break;
  case 913: imp_ana(t913,d913,ipos,liste_ana); break;
  case 914: imp_ana(t914,d914,ipos,liste_ana); break;
  case 915: imp_ana(t915,d915,ipos,liste_ana); break;
  default: break;
  }
}


/* AFFICHE LES NUMERO D'ANALYSE CORRESPONDANT A UN CERTAIN CODE */
int imp_ana(tab,dim,ipos,liste_ana)
int tab[],dim,*ipos,*liste_ana;
{int i,*pt;
 pt = tab;
 blanc(*ipos);
 *ipos = (*ipos==1?max_ana:--(*ipos));
 for (i=0;i<dim;i++)
  {*ipos = (*ipos==max_ana?1:++(*ipos));
   liste_ana[*ipos] = *pt;
   posit(*ipos);
   printf(" %d",*(pt++));
  }
}
```

```
/* PROGRAMME PRINCIPAL */
main()
(int i,j,newf;
initw(t900,"1,2,3,4,5,74");
initw(t901,"96,97,98,99,102,103,105");
initw(t902,"113,328,88,94");
initw(t903,"124,119,125,250,117");
initw(t904,"265,268,269,282,283,285,286,292");
initw(t905,"271,289,296,281,279,288,293,299,290,216,287,266");
initw(t906,"217,218,219,220,222,223,225,233,236");
initw(t907,"275,300,301,302,303");
initw(t908,"165,162,164,166");
initw(t909,"198,200,202,166");
initw(t910,"198,200,202,166");
initw(t911,"198,204,533,202");
initw(t912,"531,198,204,533,202,166");
initw(t913,"2,3,4,42,18,331,198,204,533,202");
initw(t914,"442,444,123,441,443,445,446");
initw(t915,
  "18,19,26,27,63,28,321,33,35,53,30,31,8,123,441,442,443,444,445,446");

 home;
 cteos;
 imp_lan12 = oui(12,0,"Sortie simultanee du lan12",'r');
 home;
 cteos;

/* CHARGEMENT DE LA NOMENCLATURE */
if (imp_lan12)  load_nomencl();

/* INSTAURE DES PAGES D'IMPRIMANTE DE 96 LIGNES */
 if (imp_lan12)
(lptcar(27);
 lptcar(54);
 lptcar(57);
 lptcar(54);}

if (imp_lan12) ( lptnl();lptnl();lptnl();lptnl();lptnl();lptnl();
                 lptcar(14);
                 lptrepeat(17,' ');
                 lptstr("LAN12");
                 lptnl();lptnl();lptnl();lptnl();lptnl();
                 lptrepeat(11,' ');
                 lptdate(0);
                 lptnl();
                 lptcar(15);
                 lptcar(next_page);    /* SAUT DE PAGE */
               }
```

```
/* OUVERTURE DU FICHIER FILREF */
 if ((filref = open ("b:filref.dat",2)) == -1)
               {home; cteos;
                pos(10,10);printf("On recommence une nouvelle journee :'
                printf("fichiers remis a zero !");
                if (! oui(12,40,"confirmation",'r')) {home; cteos;exit(
                filref = creat ("b:filref.dat");
                nb_ref = nro_sect_ref = 0;
                setmem(out_ref_buffer,128,'\0');
                write (filref,out_ref_buffer,1);
                seek (filref,nro_sect_ref,0);
               }
        else   {seek (filref,-1,2);
                read (filref,out_ref_buffer,1);
                seek (filref,-1,2);
                ptr_buf = out_ref_buffer;
                nro_sect_ref = tell(filref) ;
                for (nb_ref = 0;isupper((ptr_buf++)->lettre);nb_ref++);
               }
 if (filref == -1)   {msg_err(errmsg(errno())); exit();  }

/* OUVERTURE DU FICHIER FILDDE */
 if ((fildde = open ("b:fildde.dat",2)) == -1)
                fildde = creat ("b:fildde.dat");
        else    seek (fildde,0,2);
 if (fildde == -1)   {msg_err(errmsg(errno())); exit();  }

/* OUVERTURE DU FICHIER FILVAL */
 filval = open ("a:filval.dat",2);
 if (filval == -1)   {msg_err(errmsg(errno())); exit();  }
 load_val_ref(tab_val_cour);

/* OUVERTURE DU FICHIER TRVLST */
if (fopen("b:trvlst.dat",trvlst) == -1) fcreat("b:trvlst.dat",trvlst);
   else {fclose(trvlst);
         unlink ("b:trvlst.bck");
         rename("b:trvlst.dat","b:trvlst.bck");
         fopen("b:trvlst.bck",trvbck);
         fcreat("b:trvlst.dat",trvlst);
         while(fgets(tmps,trvbck) != 0) fputs(tmps,trvlst);
         fclose(trvbck);
        }

/*CHARGEMENT DES NUMEROS D'ANALYSE POUR LESQUELS IL NE FAUT PAS DE LISTE
fopen("a:fnolst.dat",iobuf);
setmem(swlst,1024,'o');
for(;fscanf(iobuf,"%d\n",&nn) > 0;)  swlst[nn] = 'n';
fclose (iobuf);
```

```
/* FIN DES INITIALISATIONS ET DEBUT DU TRAITEMENT */
 lettre_cour = 'J';
 refnb = 1;
debut:
 for (i=0;i <= (max_ana) ; i++)  lst_ana[i] = -1;
 map_vide();
 for (;! trt_ref( ););
 for (i=0;i <= 63; i++) dde[i] = 0;
 tab_write(lst_ana);
 if (get_tab(lst_ana) == -1) goto debut;
 if (ref_existante)
  genlist = oui(23,0,"generation des listes de travail",'N');
  else genlist = 1;
 if (imp_lan12)
  { lptnl();
    if (ref_existante)  lptstr(ref_select);
        else    {       concat_ref (ref_select);
                        ref_select[6] = '\0';
                        lptstr(ref_select);
                }

   lptnl();
   lptstr("------");
   lptnl();
  }
 for (i=1;i <= max_ana ; i++)
        if ((lst_ana[i] > -1) && (lst_ana[i] <= 1023))
                                { setdde(dde,lst_ana[i]);
                                  if (genlist) enreg(lst_ana[i]);
                                   if (imp_lan12) (lptint(lst_ana[i]);
                                                   lptcar(' ');
                                                   lptstr(
                                                    nomencla[lst_ana[i]]);
                                                   lptnl();
                                                  }
                                }
```

```
/* SAUVETAGE DE LA REFERENCE SUR FICHIER */
 if (! ref_existante)
(if (nb_ref == 16) (     setmem(out_ref_buffer,128,'\0');
                         ++nro_sect_ref;
                         nb_ref = 0;
                         gen_save();
                    }
 out_ref_buffer[nb_ref].lettre = lettre_cour;
 for(j=0;j<=4;j++)
       out_ref_buffer[nb_ref].ref[j] =
                  tab_val_cour[lettre_cour - 65].nro[j];
 out_ref_buffer[nb_ref].car_return = 13;
 out_ref_buffer[nb_ref].line_feed  = 10;
 seek (filref,nro_sect_ref,0);
 if (write(filref,out_ref_buffer,1) == -1)
        msg_err("ERREUR ECRITURE REFERENCE SUR DISQUE");
 nb_ref++;
}
/* SAUVETAGE  DES  DEMANDES  SUR FICHIER */
 if (ref_existante) (seek (fildde,pos_ref,0);
                     write(fildde,dde,1);)
        else        (seek (fildde,0,2);
                     if(write(fildde,dde,1) == -1)
                        msg_err("ERREUR ECRITURE DEMANDE SUR DISQUE");
                    }

/* INCREMENTATION DE LA REFERENCE UTILISEE */
if ( ! ref_existante)  ( plus1(tab_val_cour[lettre_cour - 65]);
                         save_val_ref();
                         refnb++;
                       }
 goto debut;
}
```

10.4 Le programme d'impression des petites listes.

10.4.1 Signification des principales variables.

liste	table contenant toutes les lignes du fichier TRVLST
nomencla	table des noms d'analyse

10.4.2 Description des fonctions.

load_nomencl() charge les nom d'analyse dans nomencla

10.4.3 Programme principal.

- chargement de la nomenclature des analyses
- chargement des lignes de TRVLST dans la table liste
- tri de la table liste pour regrouper les lignes par analyse.
- Impression de la table liste avec une rupture (saut de page + entête

quand le numéro d'analyse d'une ligne est différent de celui de la précédente.

-fermeture des fichiers et changement du nom de TRVLST.DAT en TRVLST.Hnn où nn est l'heure à laquelle se déroule le programme.

10.4.4 Utilisation.

écran 1

PETITES LISTES DE TRAVAIL.

369 LIGNES CHARGEES.
369 LIGNES TRIEES.

IMPRESSION EN COURS .....

2/ 5/84 11:50:14

0 S.VITESSE DE SEDIMENTAT.

J59462
J59463
J59465
J59466
J59467
J59468
J59469
J59470
J59473
J59475
J59480
J59497
J59501
J59503
J59506
J59508
J59509
J59510
J59513
J59521

2/ 5/84 11:50:20

20 S.THYMOL

J59503
J59544
J59547
J59548
J59549

2/ 5/84 11:50:25

106 S.LITHIUM PLASMATIQUE

J59492

2/ 5/84 11:50:25

144 U.PROTEINES TOTALES

J59468
J59507
J59519
J59521
J59522
J59524
J59527
J59533
J59536
J59537
J59538
J59539
J59543
J59547
J59549
J59551
J59556
J59558
J59559
J59562

10.4.5 Texte source.

```
/* IMPRESSION DES PETITES LISTES DE TRAVAIL */
#define next_page  12
#include "a:bdscio.h"
#include "a:screen.c"
#define dimliste 1750

char iobuf[BUFSIZ];
char trvlst[BUFSIZ];

struct {char str[11];} liste[dimliste],*ptliste;

char prec[11],tmp[11];

char bid[13];

int n,nb_str,i,j,strcmp();

struct  { char lib[26]; } nomencla[850];

int taille_nomencla,cpt;

struct nro_ref {char nro[6];};


/* CHARGE DANS "NOMENCLA" LA LISTE DES NOMS D'ANALYSES */
load_nomencl()
(int i,n,x;
 char tmp[26];
 fopen ("a:fanal.dat",iobuf);
 for (n=0 ;fscanf(iobuf,"%d %s\n",&x,tmp)>0;n++ )
        (for (;x != n;)
                strcpy(nomencla[n++].lib,"**************************");
         strcpy(nomencla[n],tmp);
        }
        taille_nomencla = --n;
 fclose(iobuf);
}
```

```
main()
{int z,reste;
 int heure;
 char zone[3];
 char newname[14];
 lptcar(next_page);    /* SAUT DE PAGE */

cadre();
pos(3,25);
printf("PETITES LISTES DE TRAVAIL.  ");

/* DEFINIT DES PAGES D'IMPRIMANTE DE 48 LIGNES */
lptcar(27);
lptcar(54);
lptcar(52);
lptcar(56);

/* CHARGEMENT DES NOMS ET NUMEROS D'ANALYSE */
load_nomencl();


/* OUVERTURE DU FICHIER TRVLST*/
fopen ("b:trvlst.dat",trvlst);

/* CHARGEMENT DES STR DE TRVLST  */
nb_str  = 0;
ptliste = liste;
while ((fgets(bid,trvlst) > 0) && (nb_str <dimliste))
                              { bid[10] = '\0';
                                strcpy(*(ptliste++),bid);
                                nb_str++;
                              }
pos(5,25);
printf("%d LIGNES CHARGEES.\n",nb_str);
qsort(liste,nb_str,11,strcmp);
pos(6,25);
printf("%d LIGNES TRIEES  .\n",nb_str);
```

```
/* BOUCLE D'IMPRESSION */
 pos(10,25);
 printf("IMPRESSION EN COURS ...  ");
 strcpy(prec,"9999xxxxxx");
 for (i=0;i<nb_str;i++)
        (if (strcmp(prec,liste[i].str) != 0)
                (if (egaux(prec,liste[i].str,4) != 1) cpt = 99;
                 sscanf(liste[i].str,"%d%s\n",&n,tmp);
                 if (cpt > 20)
                (lptcar(next_page);
                 lptdate(0);
                 lptnl();
                 lptnl();
                 lptint(n);
                 lptstr(" ");
                 lptstr(nomencla[n]);
                 lptnl();
                 lptrepeat(30,'-');
                 lptnl();
                 lptnl();
                 cpt = 1;
                }
                lptstr(tmp);
                cpt++;
                lptnl(); lptnl();
                }
                strcpy(prec,liste[i]);
}
lptcar(next_page);    /* SAUT DE PAGE */

/* RESTAURE DES PAGES D'IMPRIMANTE DE 96 LIGNES */
lptcar(27);
lptcar(54);
lptcar(57);
lptcar(54);


 fclose(trvlst);
 heure = peek(call(61497,0,0,0,0)+3);
 sprintf(zone,"%d",heure);
 strcpy(newname,"b:trvlst.h");
 strcat(newname,zone);
if (rename("b:trvlst.dat",newname) == -1) msg_err("erreur");
for (i=1;i<=60;i++) bell;
home;
cteos;
printf("OK");
}
```

10.5 Le programme d'impression des tableaux.

10.5.1 Signification des principales variables.

Dans tout le texte qui suit, "*****" sera mis pour un des mot suivants: cobas, sma12, bact, hémo, hvirale, thyro, sexu, bsex, urine, migra, immu, neph, rein, onco, anémie, gros, diad, medi, bw, toxo qui sont les noms des différents tableaux crées.

Parmi ces tableaux, on distingue:

1. des tableaux particuliers: ils nécessitent un traitement et/ou une impression particulière; il s'agit des tableaux urine, cobas, sma12 et bactériologie.

2. des tableaux standards: à peu de choses près, ils ont le même format et sont générés par le méme traitement. C'est donc deux procédures paramétrables qui se chargent du traitement et de l'impression.

Pour chaque patient (référence), on regarde s'il doit figurer dans une ligne du tableau *****, il y figurera si au moins une des analyses de ce tableau est demandée pour ce patient.

Les tableaux sont memorisés sous forme de ligne.

1. les lignes des tableaux particuliers ont la structure suivante

   - ⟨référence⟩⟨n caracteres⟩

   le n dépendant du nombre de colonnes (= d'analyses)

dans ce tableau . Si le n ième caractère est un blanc alors il faut faire l'analyse correspondant à la n ième colonne pour la référence de cette ligne; s'il vaut "/", il ne faut pas faire cette analyse. Les lignes de chaque tableau particulier sont mémorisées dans une table particulière à ce tableau.

les lignes des tableaux standard ont la structure suivante:

- ⟨code⟩⟨référence⟩⟨8 caracteres⟩

Toutes les lignes des tableaux standards sont mémorisées dans une seule table pour des raisons d'écomnomie de place (Cfr 4.3.3); ⟨code⟩ va donc servir à déterminer à quel tableau appartient

cette ligne au moment de l'imprimer. <référence> est la référence correspondant à cette ligne. Les 8 caractères ont la même fonction que plus haut mais suivant le tableau, seuls les dim_***** sont utilises.

dim_***** nombre de colonnes dans le tableau *****

cd_***** code d'une ligne du tableau *****

tab_***** table contenant les dim_***** numéros d'analyse correspondant aux colonnes du tableau *****

10.5.2 Description des fonctions.

int tiret(a,b) Imprime "a" blancs suivis de "b" tirets

int trt_cobas() traitement particulier de la référence en cours pour le tableau cobas

int trt_sma12() traitement particulier de la référence en cours pour le tableau sma12

int trt_urine() traitement particulier de la référence en cours pour le tableau urine

int trt_bact() traitement particulier de la référence en cours pour le tableau bactério

trt_gen (tab_gen,dim_gen,code_liste)
traitement de la référence en cours pour le tableau standard tab_gen de dim_gen colonnes et de code code_liste

int new_line(n) initialise une ligne du tableau bactério avec des "/" et positionne n a 1.

int trt_carteb()génère les cartes de demande pour la référence en cours.

imp_sma12() Impression particuliere du tableau sma12

int imp_line(dim,ncar,ref_en_fin)
imprime une ligne de tableau standard dont dim colonnes sont utilisées, chacune ayant une largeur de ncar caractères et la référence est inscrite a la droite de la ligne si ref_en_fin = 1

int imp_gen(dim_tab, code_tab, s1, s2, s3, lgr_tiret, lgr_col, double_ref)

procède à l'impression du tableau standard de dimension dim_tab, de code code_tab, avec les titres s1, s2, s3; les lignes du tableau sont separées par des lignes de tirets d'une longueur lgr_tiret, la largeur des colonnes est de lgr_col et la référence est redoublée en fin de ligne si double_ref =1)

int imp_*****() Cette routine représente un simple appel a la routine précédente en lui donnant les valeurs des paramètres pour le tableau *****. ***** représente ici uniquement les tableaux standards.

int imp_migra() Impression particulière du tableau migration

int imp_urine() Impression particulière du tableau urine

int imp_cobas() Impression particulière du tableau cobas

int imp_bact() Impression particulière du tableau bactério

int charger_ref()
Chargement en mémoire des références à traiter

int entete_list(st)
Imprime l'entête d'un tableau standard en le particularisant avec le texte st.

10.5.3 Programme principal.

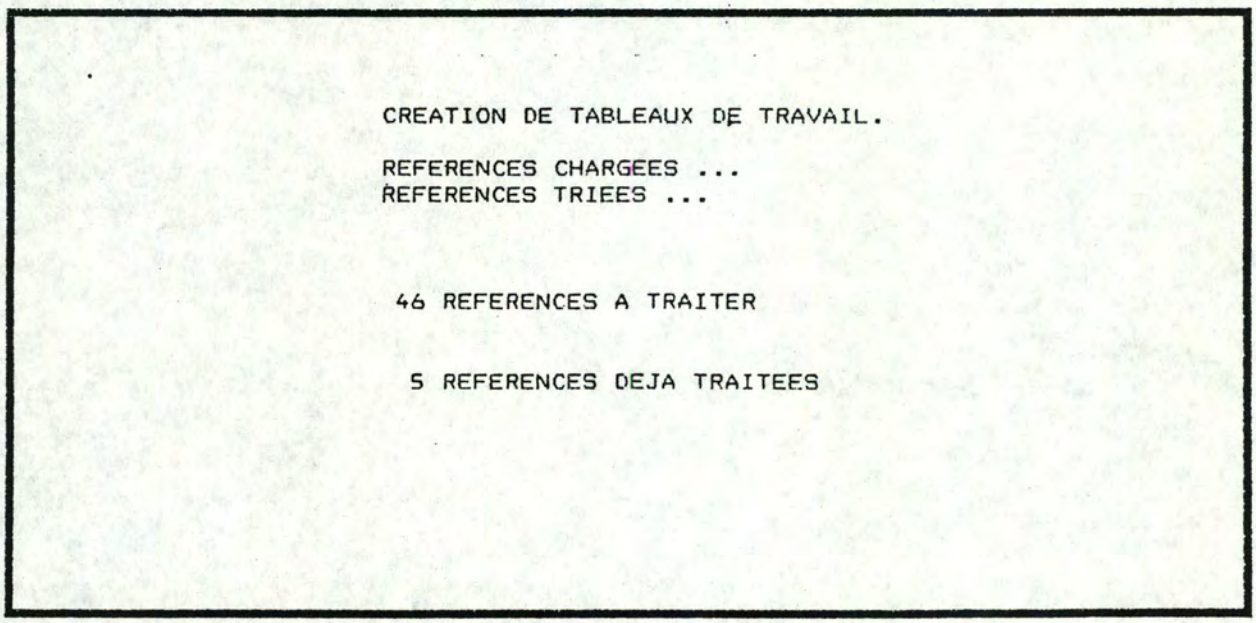
- Chargement de différentes valeurs se trouvant sur fichier et servant à imprimer certains tableaux: chaque tableau cobas ne peut contenir plus de 24 lignes et doit avoir un numéro différent du tableau précédent durant une journée. Le numéro du dernier tableau est mémorisé pour la prochaine exécution du programme; de même toute les lignes du tableau sma12 doivent être numérotées de 1 a N durant la journée. Le numéro de la dernière ligne imprimée est également mémorisé sur fichier. Enfin comme le traitement s'effectue sur le fichier FILREF qui contient toute les références de la journée, chaque traitement doit seulement s'effectuer à partir de la dernière référence traitée lors de la précédente exécution. Cette référence est également mémorisée sur fichier.

- initialisation des tables de numéros d'analyse pour chaque tableau

- chargement et tri des références pour qu'elles apparaissent triées dans les tableaux
- pour chaque référence effectuer le traitement pour chacun des tableaux
- tri sur ⟨code⟩ des lignes de tableaux standards pour les regrouper par code
- impression des tableaux particuliers et standards
- sauvetage des valeurs particulières et fermeture des fichiers

10.5.4 Utilisation.

écran 1



L I S T E 3

cobas 2/ 5/84 11:47: 7

PLATEAU NR0 5

CS	KON.L	26 GOT	27 GPT	33 LDH	41 CPK	63 GGT	28 PAL	52 LAP	40 LIP	34 AMY	321 SNU	29 PAC	105 MGS	327 MGE	49 OC
1	J59461			///	///		///	///	///	///	///	///	///	///	///
2	J59462			///	///			///	///	///	///	///	///	///	
3	J59463	///	///	///	///	///	///	///	///	///	///	///	///	///	///
4	J59464			///	///			///	///		///	///	///	///	///
5	J59465										///	///	///	///	
6	J59466	///	///	///	///	///	///	///	///	///	///	///		///	///
7	J59467	///	///	///	///		///	///	///	///	///	///			///
8	J59468			///	///		///	///	///	///	///	///		///	///
9	J59471			///	///	///	///	///	///	///	///	///	///	///	///
10	J59472			///	///	///	///	///	///	///	///	///	///	///	///
11	J59475			///	///			///	///	///	///	///	///	///	///
12	J59480										///		///	///	///
13	J59495	///	///	///	///	///	///	///	///	///	///	///			///
14	J59497			///	///	///	///	///	///	///	///	///	///	///	///
15	J59501			///	///			///	///	///	///	///		///	///
16	J59503				///			///	///	///	///		///	///	///
17	J59506				///			///		///					///
18	J59508			///				///	///	///	///	///	///	///	///
19	J59509			///	///			///	///	///	///	///	///	///	///
20	J59510			///	///			///	///	///	///	///	///	///	///
21	J59513			///	///			///	///	///	///	///	///	///	///
22	J59520				///			///	///	///	///	///	///	///	///
23	J59521	///	///	///	///	///		///	///	///	///	///	///	///	///
24	K LP														

L I S T E 3

Oncologie 2/ 5/84 11:47:32

	452 a-F.P.	435 CEA	358 B2micro	50 PAP
J59462	///	///	///	
J59466	///		///	///
J59480	///		///	
J59508			///	///
J59527	///	///	///	
J59538	///	///	///	
J59549			///	
J59551			///	
J59553			///	///
J59559		///	///	///

L I S T E 3

urines / microscopie 2/ 5/84 11:47:34

	12 globules blancs	22 globules rouges	32 cellules epithelial	42 mucus	46 oxal	50 phosp amorp	54 phosp am-mg
J59468							
J59486							
J59507							
J59519							
J59521							
J59522							
J59524							
J59527							
J59533							
J59536							
J59537							
J59538							
J59539							

L I S T E 3

Nephelometre Technicon 2/ 5/84 11:47:27

	53 CAP. FER	545 TRANSFER.	535 IgG	536 IgA	537 IgM
J59461		///////////	///////////	///////////	///////////
J59467	///////////		///////////	///////////	///////////
J59471		///////////	///////////	///////////	///////////
J59472		///////////	///////////	///////////	///////////
J59473	///////////	///////////	///////////		///////////
J59480			///////////	///////////	///////////
J59525		///////////	///////////	///////////	///////////
J59546		///////////	///////////	///////////	///////////
J59554	///////////	///////////			///////////
J59556		///////////	///////////	///////////	///////////
J59559		///////////	///////////	///////////	///////////
J59561		///////////	///////////	///////////	///////////

L I S T E 3

Thyroide 2/ 5/84 11:47:26

	355 T4 TOT	401 T3	402 TSH	404 RU	406 TBG	337 T4 LIB	356 T4 LIEE
J59461				///////	///////	///////	///////
J59467				///////		///////	///////
J59468				///////	///////	///////	///////
J59471				///////	///////	///////	///////
J59480					///////	///////	///////
J59525		///////	///////	///////	///////	///////	///////
J59553					///////	///////	///////
Z65880				///////	///////	///////	///////

L I S T E 3

Sexuelles 1 2/ 5/84 11:47:26

	346 HPL	411 OESTRI.	410 OESTRA.	414 PROGES.	416 LH	417 FSH	415 TESTOS.	418 PROLAC
J59467	////////				////////	////////	////////	////////
J59559	////////							

L I S T E 3

Hepatite virale 2/ 5/84 11:47:25

	123 Ag HBS	441 Ac HBS	443 Ac HBC	442 HEP.A	444 IGM H.A	445 AG HBe	446 AC HBe
J59465							
J59503							
J59559			////////	////////	////////		////////
J59560		////////		////////	////////		////////
J59561		////////		////////	////////		////////

10.5.5 Texte source.

```
/* GENERATION DE TABLEAUX DE TRAVAIL */
#include "a:bdscio.h"
#include "a:screen.c"
#define dim_cobas 16
#define dim_sma12 12
#define dim_bact  32
#define dim_hemo  8
#define dim_hvirale 7
#define dim_thyro 7
#define dim_sexu  8
#define dim_urine 1
#define dim_bsex  6
#define dim_migra 4
#define dim_immu  7
#define dim_neph  5
#define dim_rein  4
#define dim_onco  4
#define dim_anemie 4
#define dim_gross  3
#define dim_diab   3
#define dim_medi   2
#define dim_bw     4
#define dim_toxo   5
#define cd_hemo     'A'
#define cd_hvirale  'B'
#define cd_thyro    'C'
#define cd_sexu     'D'
#define cd_bsex     'E'
#define cd_neph     'F'
#define cd_immu     'G'
#define cd_migra    'H'
#define cd_rein     'I'
#define cd_onco     'J'
#define cd_anemie   'K'
#define cd_gross    'L'
#define cd_diab     'M'
#define cd_medi     'N'
#define cd_bw       'O'
#define cd_toxo     'P'
#define max_dim 8
#define dim_liste 150
```

```
char carteb[BUFSIZ],filref[BUFSIZ],lastref[BUFSIZ],cobas[BUFSIZ];
char sma12[BUFSIZ],nrolst[BUFSIZ];

int fildde,nro_plat,it,ana_min,ana_max,nro_sma;
int dde[64],ind_ref,nb_ref,numlist,strcmp();

int ind_sma12 ,tab_sma12[dim_sma12];
int ind_cobas ,tab_cobas[dim_cobas];
int ind_bact,   tab_bact[dim_bact];
int tab_bsex[dim_bsex];
int tab_migra[dim_migra];
int tab_immu[dim_immu];
int tab_hvirale[dim_hvirale];
int tab_thyro[dim_thyro];
int tab_sexu[dim_sexu];
int tab_neph[dim_neph];
int tab_hemo[dim_hemo];
int tab_rein[dim_rein];
int tab_onco[dim_onco];
int tab_anemie[dim_anemie];
int tab_gross[dim_gross];
int tab_diab[dim_diab];
int tab_medi[dim_medi];
int tab_bw[dim_bw];
int tab_toxo[dim_toxo];
int ind_urine;
int ind_tab,nb_line,i;


struct nro_ref { char nro[7];};

struct { struct nro_ref reff;
         char c[dim_cobas];} lg_cobas[80];

struct { struct nro_ref reff;
         char c[dim_sma12];} lg_sma12[80];

struct { struct nro_ref reff;
         char c[dim_bact];} lg_bact[50];

struct { char cd ;
         struct nro_ref refgen;
         char cgen[max_dim];} tab_lg[300];

struct { struct nro_ref reff;} lg_urine[40];

struct { struct nro_ref reff;
         int ndde;} liste[dim_liste];
```

```
/* IMPRIME a BLANCS SUIVIS DE b TIRETS */
int tiret(a,b)
int a,b;
{lptrepeat(a,' ');
 lptrepeat(b,'-');
 lptnl();
}

/* TRAITEMENT POUR LES TABLEAUX COBAS */
int trt_cobas()
{int deja_ref,i;
deja_ref = 0;
for (i=0;i<dim_cobas;i++)
        if (testdde(dde,tab_cobas[i]))
                {if (! deja_ref)
                   {strcpy(lg_cobas[++ind_cobas].reff,
                                liste[ind_ref].reff);
                    setmem(lg_cobas[ind_cobas].c,dim_cobas,'/');
                    deja_ref = 1;
                    }
                 lg_cobas[ind_cobas].c[i] = ' ';
                }
}

/* TRAITEMENT POUR LES TABLEAUX SMA12 */
int trt_sma12()
{int deja_ref,i;
deja_ref = 0;
for (i=0;i<dim_sma12;i++)
        if (testdde(dde,tab_sma12[i]))
                {if (! deja_ref)
                   {strcpy(lg_sma12[++ind_sma12].reff,
                                liste[ind_ref].reff);
                    setmem(lg_sma12[ind_sma12].c,dim_sma12,'/');
                    deja_ref = 1;
                    }
                 lg_sma12[ind_sma12].c[i] = ' ';
                }
}
```

```
/* TRAITEMENT  D'UN  TABLEAUX STANDARD  */
int trt_gen(tab_gen,dim_gen,code_liste)
int tab_gen[];
int dim_gen;
char code_liste;

{int deja_ref,i;
deja_ref = 0;
for (i=0;i<dim_gen;i++)
        if (testdde(dde,tab_gen[i]))
                {if (! deja_ref)
                    {strcpy(tab_lg[++ind_tab].refgen,liste[ind_ref].reff
                     tab_lg[ind_tab].cd = code_liste;
                     setmem(tab_lg[ind_tab].cgen,dim_gen,'/');
                     deja_ref = 1;
                    }
                 tab_lg[ind_tab].cgen[i] = ' ';
                }
}



/* TRAITEMENT POUR LES TABLEAUX URINE */
int trt_urine()
{if (testdde(dde,146))
        strcpy(lg_urine[++ind_urine].reff,liste[ind_ref].reff);

}


/* INITIALISE UNE NOUVELLE LIGNE POUR LES TABLEAUX BACTERIOLOGIE */
int  new_line(n)
int *n;
{strcpy(lg_bact[++ind_bact].reff,liste[ind_ref].reff);
 setmem(lg_bact[ind_bact].c,dim_bact,'/');
 *n = 1;
}
```

```
/* TRAITEMENT POUR LES TABLEAUX BACTERIOLOGIE */
int trt_bact()
(int deja_ref,i;
deja_ref = 0;
for (i=0;i(dim_bact;i++)
        if (testdde(dde,tab_bact[i]))
                (if (! deja_ref) new_line(&deja_ref);
                 lg_bact[ind_bact].c[i] = ' ';)
if ((testdde(dde,299)) uu (testdde(dde,225)))
        (if (! deja_ref) new_line(&deja_ref);
         lg_bact[ind_bact].c[1] = ' ';)
if ((testdde(dde,290)) uu(testdde(dde,233)))
        (if (! deja_ref) new_line(&deja_ref);
         lg_bact[ind_bact].c[2] = ' ';
         lg_bact[ind_bact].c[3] = ' ';
         lg_bact[ind_bact].c[4] = ' ';
         lg_bact[ind_bact].c[5] = ' ';)
if (testdde(dde,216))
        (if (! deja_ref) new_line(&deja_ref);
         lg_bact[ind_bact].c[7] = ' ';)
if (testdde(dde,271))
        (if (! deja_ref) new_line(&deja_ref);
         lg_bact[ind_bact].c[9] = ' ';)
if (testdde(dde,279))
        (if (! deja_ref) new_line(&deja_ref);
         lg_bact[ind_bact].c[10] = ' ';)
if (testdde(dde,280))
        (if (! deja_ref) new_line(&deja_ref);
         lg_bact[ind_bact].c[10] = ' ';)
if (testdde(dde,281))
        (if (! deja_ref) new_line(&deja_ref);
         lg_bact[ind_bact].c[12] = ' ';)
if (testdde(dde,268))
        (if (! deja_ref) new_line(&deja_ref);
         lg_bact[ind_bact].c[13] = ' ';)
if (testdde(dde,219))
        (if (! deja_ref) new_line(&deja_ref);
         lg_bact[ind_bact].c[15] = ' ';)
if (testdde(dde,293))
        (if (! deja_ref) new_line(&deja_ref);
         lg_bact[ind_bact].c[25] = ' ';)
)
```

```
/* CREATION DES CARTES DE DEMANDES */
int trt_carteb ()
(int cpt,i,*pt;
ana_min =0;
while ((dde[ana_min] == 0) && (ana_min <64)) ana_min++;
ana_min = ana_min * 16;
pt = dde[63];
for (ana_max = 64; *(pt--) == 0; ana_max--);
ana_max = (ana_max * 16) -1;
cpt = 19;
for (i=ana_min;i<= ana_max;i++)
        if (testdde(dde,i))
                (if (cpt>18)
                        (fprintf(carteb,"\n");
                         putc('B',carteb);
                         fputs(liste[ind_ref].reff,carteb);
                         cpt = 1;
                        }
                 cpt++;
                 fprintf(carteb,"%4.4d",i);
                }
}

/* IMPRESSION DU TABLEAU SMA12 */
int imp_sma12()
(int i,j,cpt;
cpt = 99;
for (i=0;i<=ind_sma12;i++)
 (if (cpt>25)
   (entete_list("SMA12  ");
    lptrepeat(13,' ');
    lptstr("  96   97   99   35   18   102  103   8   42   36   30   31'
    lptnl();
    lptrepeat(13,' ');
   lptstr("  NA   K   R.AL  FE  P.TO  CA   P    CH  AC.U CREA BTOT BDIR'
    lptnl();     tiret(13,60);     cpt = 1;
   }
   lptint(++nro_sma);
   lptcar(' ');
   lptstr(lg_sma12[i].reff);
   lptstr(" u");
   for(j=0;j <= dim_sma12-1;j++)( lptrepeat(4,lg_sma12[i].c[j]);
                                  lptcar('u');}
   lptnl();   tiret(13,60);   cpt++;
 }
}
```

```
/* IMPRESSION D'UNE LIGNE DE TABLEAU STANDARD */
int imp_line(dim,ncar,ref_en_fin)
int dim,ncar,ref_en_fin;
  {int j;
   lptstr(tab_lg[ind_tab].refgen);
   lptstr(" u");
   for(j=0;j <= dim-1;j++){ lptrepeat(ncar,tab_lg[ind_tab].cgen[j]);
                                    lptcar('u');}
   if (ref_en_fin) {lptcar(' ');
                    lptstr(tab_lg[ind_tab].refgen);}
   lptnl();
  }


/* IMPRESSION D'UN TABLEAU STANDARD */
int imp_gen(dim_tab,code_tab,s1,s2,s3,lgr_tiret,lgr_col,double_ref)
int dim_tab;
char code_tab,*s1,*s2,*s3;
int lgr_tiret,lgr_col,double_ref;
{int i,j,cpt;
cpt = 99;
while ((tab_lg[ind_tab].cd == code_tab ) && (ind_tab <= nb_line))
 {if (cpt>60)
   {entete_list(s1);
    lptrepeat(7,' ');
    lptstr(s2);
    lptnl();
    lptrepeat(7,' ');
    lptstr(s3);
    lptnl();
    tiret(7,lgr_tiret);
    cpt = 1;
   }
   imp_line(dim_tab,lgr_col,double_ref);
   tiret(7,lgr_tiret);
   cpt++;
   ind_tab++;
 }
}
```

```
/* IMPRESSION DU TABLEAU HEMATOLOGIE */
int imp_hemo()
{imp_gen(dim_hemo,cd_hemo,
     "Hemostase  ",
     "   69      72      314     315     71      73      75      324",
     " T.PRO.A THROM.A T.PRO.E THROM.E T.CEPH. T.HOWEL FIBRIN. T.THROM"
     64,7,1);
}

/* IMPRESSION DU TABLEAU HEPATITE VIRALE */
int imp_hvirale()
{imp_gen(dim_hvirale,cd_hvirale,
     "Hepatite virale  ",
     "  123     441      443    442     444     445     446",
     " Ag HBS  Ac HBS  Ac HBC  HEP.A   IGM H.A AG HBe  AC HBe",
     56,7,0);
}

/* IMPRESSION DU TABLEAU THYROIDE */
int imp_thyro()
{imp_gen(dim_thyro,cd_thyro,
     "Thyroide  ",
     "   355     401      402    404      406   337     356",
     " T4 TOT     T3       TSH     RU      TBG  T4 LIB  T4 LIEE",
     56,7,0);
}

/* IMPRESSION DU TABLEAU SEXUELLES1 */
int imp_sexu()
{imp_gen(dim_sexu,cd_sexu,
     "Sexuelles 1 ",
     "   346     411    410      414     416     417    415     418",
     "   HPL   OESTRI. OESTRA. PROGES.   LH      FSH   TESTOS. PROLAC",
     64,7,0);
}

/* IMPRESSION DU TABLEAU SEXUELLES2 */
int imp_bsex()
{imp_gen(dim_bsex,cd_bsex,
     "Sexuelles 2 ",
     "    412        425        426        428        438         434",
     "  OESTRONE   17 OH PR.   ANDROS.   DHEA SULF.   DHT       SEX B.G."
     67,10,0);
}
```

```
/* IMPRESSION DU TABLEAU NEPHELOMETRE TECHNICON */
int imp_neph()
{imp_gen(dim_neph,cd_neph,
     "Nephelometre Technicon ",
     "   53          545         535         536         537",
     "CAP. FER    TRANSFER.     IgG         IgA         IgM",
     55,10,0);
}

/* IMPRESSION DU TABLEAU REIN */
int imp_rein()
{imp_gen(dim_rein,cd_rein,
     "Rein ",
     "   436      431      347      349",
     " ALDOST. CORTIS.  ACTH    RENINE",
     32,7,0);
}

/* IMPRESSION DU TABLEAU ONCOLOGIE */
int imp_onco()
{imp_gen(dim_onco,cd_onco,
     "Oncologie",
     "   452      435      358      50",
     " a-F.P.     CEA    B2micro    PAP",
     32,7,0);
}

/* IMPRESSION DU TABLEAU ANEMIE */
int imp_anemie()
{imp_gen(dim_anemie,cd_anemie,
     "Anemie ",
     "   338      341      340      343",
     " FERRIT. AC.FOL. VIT.B12 A.FOL.ER",
     32,7,0);
}

/* IMPRESSION DU TABLEAU GROSSESSE */
int imp_gross()
{imp_gen(dim_gross,cd_gross,
     "Grossesse",
     "   419      420      392",
     "   HCG     HCG-B     SP1",
     24,7,0);
}
```

```
/* IMPRESSION DU TABLEAU DIABETE */
int imp_diab()
{imp_gen(dim_diab,cd_diab,
     "Diabete",
     "   334      421      422",
     " C-PEPT    INSUL.  INS.EPR.",
     24,7,0);
}


/* IMPRESSION DU TABLEAU MEDICAMENT */
int imp_medi()
{imp_gen(dim_medi,cd_medi,
     "Medicament",
     "    394        395",
     " DIGOXINE  DIGITOXINE",
     20,9,0);
}


/* IMPRESSION DU TABLEAU BILAN B.W. */
int imp_bw()
{imp_gen(dim_bw,cd_bw,
     "Bilan B.W.",
     "   328        88         94        113",
     " V.D.R.L  T.P.H.A  SYPH.IF  BW+CTRL",
     36,8,0);
}


/* IMPRESSION DU TABLEAU TOXO */
int imp_toxo()
{imp_gen(dim_toxo,cd_toxo,
     "Bilan toxo",
     "   124      119      125     117       250",
     " TOX.AGG TOX.HEM TOX.IgG TOX.IgM TOX.FC.",
     40,7,0);
}



/* IMPRESSION DU TABLEAU IMMUNO-PROTEINES */
int imp_immu()
{imp_gen(dim_immu,cd_immu,
     "Immuno proteines  ",
     "   541       542       550       551        62       543       544",
     "   CER    ALPHA2 M. COMP C3  COMP C4    OROSO  ALPHA1 AT    HPT",
     64,8,0);
}
```

```
/* IMPRESSION DU TABLEAU MIGRATIONS */
int imp_migra()
(int i,j,cpt;
cpt = 99;
while ((tab_lg[ind_tab].cd == cd_migra ) && (ind_tab <= nb_line))
 (if (cpt>60)
   (entete_list("     migration  ");
    lptrepeat(7,' ');
    lptstr("       19              11              16              17")
    lptnl();
    lptrepeat(7,' ');
    lptstr(
    "uPROT. SERIQUES u    LIPIDO     u  CHOL. LIPO.  u   ISO. LDH.   u"
    lptnl();
    tiret(7,64);
    cpt = 1;
   }
   lptstr(tab_lg[ind_tab].refgen);
   lptstr(" u");
   for(j=0;j <= dim_migra-1;j++)( lptrepeat(7,tab_lg[ind_tab].cgen[j]);
                                  lptcar('*');
                                  lptrepeat(7,tab_lg[ind_tab].cgen[j]);
                                  lptcar('u');}
   lptnl();
   tiret(7,64);
   cpt++;
   ind_tab++;
 }
}
```

```
/* IMPRESSION DU TABLEAU URINES */
int imp_urine()
(int i,j,cpt;
cpt = 99;
for (i=0;i<=ind_urine;i++)
 (if (cpt>30)
   (entete_list("urines / microscopie   ");
    lptrepeat(7,' ');
    lptstr(" 12          22          32          42    46    50    54");
    lptnl();
    tiret(7,70);
    lptrepeat(7,' ');
    lptstr("u globules u globules u cellules umucusuoxal uphospuphospu");
    lptnl();
    lptrepeat(7,' ');
    lptstr("u  blancs  u  rouges  uepithelialu     u     uamorpuam-mgu");
    lptnl();
    tiret(7,70);
    cpt = 1;
   }
   lptstr(lg_urine[i].reff);
   lptstr(" u           u           u           u     u     u     u     u");
   lptnl();
   lptrepeat(7,' ');
   lptstr("u           u           u           u     u     u     u     u");
   lptnl();
   tiret(7,70);
   cpt++;
 }
}
```

```
/* IMPRESSION DU TABLEAU COBAS */
int imp_cobas()
{int i,j,cpt;
cpt = 99;
for (i=0;i<=ind_cobas;i++)
 {if (cpt>23)
   {if (cpt != 99)
   {lptstr("   24 K LP   ");
    lptstr(
     "u   u   u   u   u   u   u   u   u   u   u   u   u   u   u   u   u'
    lptnl();    tiret(13,64);
    }
    entete_list("cobas   ");
    lptrepeat(5,' ');
    lptstr("PLATEAU NRO ");
    lptint(++nro_plat);
    lptnl();   lptnl();   lptnl();
    lptrepeat(13,' ');
    lptstr(
      " 26  27  33  41  63  28  52  40  34  321 29  105 327 49  23  37":
    lptnl();
    lptrepeat(13,' ');
    lptstr(
      " GOT GPT LDH CPK GGT PAL LAP LIP AMY 5NU PAC MGS MGE OCT TRI PHO'
    lptnl();
    tiret(13,64);
    lptstr("    CS KON.L ");
    lptstr(
      "u   u   u   u   u   u   u   u   u   u   u   u   u   u   u   u   u   t

    lptnl();     tiret(13,64);     cpt = 1;
   }
   lptint(cpt);     lptcar(' ');
   lptstr(lg_cobas[i].reff);    lptstr(" u");
   for(j=0;j <= dim_cobas-1;j++){ lptrepeat(3,lg_cobas[i].c[j]);
                                  lptcar('u');}
   lptnl();     tiret(13,64);    cpt++;
 }
 if (cpt != 99)
{lptstr("   24 K LP   ");
  lptstr(
    "u   u   u   u   u   u   u   u   u   u   u   u   u   u   u   u   u")
    lptnl();    tiret(13,64);
}
}
```

```
/* IMPRESSION DU TABLEAU BACTERIOLOGIE  */
int imp_bact()
{int i,j,cpt;
cpt = 99;
for (i=0;i<=ind_bact;i++)
 {if (cpt>60)
   {entete_list("          virologie   ");
    lptrepeat(7,' ');
    lptstr("     299 290");
    lptnl();    lptrepeat(7,' ');
    lptstr(
      "     225 233                  216     271 279     281 268     219'
    lptnl();    lptrepeat(7,' ');
    lptstr(
      " 265 275 300 301 302 303     269 296 217 222 280 220 218 236 305'
    lptnl();    lptrepeat(7,' ');
    lptstr(
      " ADE  B1  B2  B3  B4  B5  B6 ENT PIC CYT  H1  H2 VAR MYC R&B CHL'
    lptnl();    tiret(7,64);    cpt = 1;
   }
   lptstr(lg_bact[i].reff);   lptstr(" u");
   for(j=0;j <= 15;j++){ lptrepeat(3,lg_bact[i].c[j]);
                         lptcar('u');}
   lptnl();   tiret(7,64);   cpt++;
 }
cpt = 99;
for (i=0;i<=ind_bact;i++)
 {if (cpt>60)
   {entete_list("          virologie   ");
    lptrepeat(7,' ');
    lptstr("                                        293");
    lptnl();    lptrepeat(7,' ');
    lptstr(
      " 306 307 282 283     285     286 288 223 292     274 308     289'
    lptnl();    lptrepeat(7,' ');
    lptstr(
      " ORN GON INa INb INc PA1 PA2 PA3 ORE ROU RSV ROT RUB  Y3  Y9 POL'
    lptnl();    tiret(7,64);    cpt = 1;
   }
   lptstr(lg_bact[i].reff);   lptstr(" u");
   for(j=16;j <= dim_bact-1;j++){ lptrepeat(3,lg_bact[i].c[j]);
                         lptcar('u');}
   lptnl();   tiret(7,64);   cpt++;
 }
}
```

```
/* CHARGEMENT DES REFERENCES DE FILREF A TRAITER EN TABLEAUX */
int charger_ref()
{int i,sect;
 struct nro_ref rf,last;
 i = -1;
 sect =  0;
 fscanf(lastref,"%s\n",last);
 if (strcmp(last,"ALL") != 0)
        {fscanf(filref,"%s\n",rf);
         while (strcmp(last,rf) != 0)
                {fscanf(filref,"%s\n",rf);
                 sect++;
                }
        }
  else sect = -1;
  while ((fscanf(filref,"%s\n",rf) > 0) && (i < dim_liste))
        {strcpy(liste[++i].reff,rf);
         liste[i].ndde = ++sect;
         if (! isupper(rf.nro[0])) {i--; sect--;}
        }
  nb_ref = i + 1;
}


/* IMPRESSION DE L'ENTETE POUR UN TABLEAU */
int entete_list(st)
char *st;
{lptcar(12);
 lptrepeat(20,' ');
 lptcar(14);
 lptstr("L I S T E  ");
 lptint(numlist);
 lptcar(15);
 lptnl();
 lptnl();
 lptstr(st);
 lptrepeat(10,' ');
 lptdate(0);
 lptnl();
 lptnl();
}
```

```
main()
{cadre();
pos(3,25);
printf("CREATION DE TABLEAUX DE TRAVAIL.  ");

lptcar(12);    /* SAUT DE PAGE */
/* INSTAURE DES PAGES D'IMPRIMANTE DE 48 LIGNES */
lptcar(27);
lptcar(54);
lptcar(52);
lptcar(56);

fopen("b:lastref.dat",lastref);
fopen("b:filref.dat",filref);
fildde = open("b:fildde.dat",0);
fcreat("a:carteb.pun",carteb);
fprintf(carteb,"CETTE CARTE DOIT ETRE ENLEVEE !!!!");
fopen("b:plateau.dat",cobas);
fscanf(cobas,"%d\n",&nro_plat);
fopen("b:nrosma.dat",sma12);
fscanf(sma12,"%d\n",&nro_sma);
fopen("b:nrolst.dat",nrolst);
fscanf(nrolst,"%d\n",&numlist);
++numlist;

initw(tab_cobas,"26,27,33,41,63,28,52,40,34,321,29,105,327,49,23,37");
ind_cobas = -1;
initw(tab_sma12,"96,97,99,35,18,102,103,8,42,36,30,31");
ind_sma12 = -1;
initw(tab_hemo,"69,72,314,315,71,73,75,324");
initw(tab_hvirale,"123,441,443,442,444,445,446");
initw(tab_thyro,"355,401,402,404,406,337,356"),
initw(tab_sexu,"346,411,410,414,416,417,415,418");
initw(tab_bsex,"412,425,426,428,438,434");
initw(tab_neph,"53,545,535,536,537");
initw(tab_immu,"541,542,550,551,62,543,544");
initw(tab_migra,"19,11,16,17");
initw(tab_rein,"436,431,347,349");
initw(tab_onco,"452,435,358,50");
initw(tab_anemie,"338,341,340,343");
initw(tab_gross,"419,420,392");
initw(tab_diab,"334,421,422");
initw(tab_medi,"394,395");
initw(tab_bw,"328,88,94,113");
initw(tab_toxo,"124,119,125,117,250");
initw(tab_bact,"265,275,300,301,302,303,1020,269,296,217,222,280,220,218
36,305,306,307,282,283,1020,285,1020,286,288,223,292,1020,274,308,1020,289");
ind_bact = -1;
ind_urine = -1;
ind_tab = -1;
```

```
charger_ref();
pos(5,25);
printf("REFERENCES CHARGEES ...");
fclose(lastref);
fcreat("b:lastref.dat",lastref);
fprintf(lastref,"%s\n",liste[nb_ref -1].reff);
qsort(liste,nb_ref,9,strcmp);
pos(6,25);
printf("REFERENCES TRIEES ...");
pos(10,25);
printf("%3.3d REFERENCES A TRAITER",nb_ref);
pos(13,25);
printf("%3.3d REFERENCES DEJA TRAITEES",0);
for(ind_ref=0;ind_ref<nb_ref;ind_ref++)
        {pos(13,25);
         printf("%3.3d",ind_ref + 1);
         seek(fildde,liste[ind_ref].ndde,0);
         read(fildde,dde,1);
         trt_cobas();
         trt_sma12();
         trt_gen(tab_hvirale,dim_hvirale,cd_hvirale);
         trt_gen(tab_immu,dim_immu,cd_immu);
         trt_gen(tab_bsex,dim_bsex,cd_bsex);
         trt_gen(tab_neph,dim_neph,cd_neph);
         trt_gen(tab_migra,dim_migra,cd_migra);
         trt_gen(tab_thyro,dim_thyro,cd_thyro);
         trt_gen(tab_sexu,dim_sexu,cd_sexu);
         trt_gen(tab_hemo,dim_hemo,cd_hemo);
         trt_gen(tab_rein,dim_rein,cd_rein);
         trt_gen(tab_onco,dim_onco,cd_onco);
         trt_gen(tab_anemie,dim_anemie,cd_anemie);
         trt_gen(tab_gross,dim_gross,cd_gross);
         trt_gen(tab_diab,dim_diab,cd_diab);
         trt_gen(tab_medi,dim_medi,cd_medi);
         trt_gen(tab_bw,dim_bw,cd_bw);
         trt_gen(tab_toxo,dim_toxo,cd_toxo);
         trt_carteb();
         trt_bact();
         trt_urine();
        }
nb_line = ind_tab + 1;
ind_tab = 0;
qsort(tab_lg,nb_line,16,strcmp);
pos(16,25);
printf("%d LIGNES ",nb_line);
```

```
pos(20,25);
printf("IMPRESSION EN COURS ...  ");

imp_cobas();
imp_sma12();
imp_bact();
imp_hemo();
imp_hvirale();
imp_thyro();
imp_sexu();
imp_bsex();
imp_neph();
imp_immu();
imp_migra();
imp_rein();
imp_onco();
imp_anemie();
imp_gross();
imp_diab();
imp_medi();
imp_bw();
imp_toxo();
imp_urine();
```

```
fclose(cobas);
fcreat("b:plateau.dat",cobas);
fprintf(cobas,"%d\n",nro_plat);
fclose(sma12);
fcreat("b:nrosma.dat",sma12);
fprintf(sma12,"%d\n",nro_sma);
fclose(nrolst);
fcreat("b:nrolst.dat",nrolst);
fprintf(nrolst,"%d\n",numlist);
fprintf(carteb,"\nDERNIERE CARTE : A ENLEVER !\n");
putc(CPMEOF,carteb);
lptcar(12);
putc(CPMEOF,cobas);
fclose(cobas);
putc(CPMEOF,sma12);
fclose(sma12);
putc(CPMEOF,nrolst);
fclose(nrolst);
close(fildde);
fclose(carteb);
fclose(filref);
putc(CPMEOF,lastref);
fclose(lastref);
pos(20,0);
for (it=1;it<=60;it++) bell
lptcar(12);    /* SAUT DE PAGE */
/* INSTAURE DES PAGES D'IMPRIMANTE DE 96 LIGNES */
lptcar(27);
lptcar(54);
lptcar(57);
lptcar(54);
home;
cteos;
printf("OK");
}
```

10.6 Le programme de récupération en cas de bourrage.

10.6.1 Signification des principales variables.

bourrage, tableaux_ok, ptliste_ok
booléens disant si oui ou non il y a eu un problème

10.6.2 Description des fonctions.

int fini() effectue la clôture du programme

10.6.3 Programme principal.

- test de l'existence et de l'accessibilité des fichiers FILDDE, FILREF, FILVAL, FNOLST, NROLST, NROSMA, PLATEAU, FANAL, LASTREF, TRVLST;
- question interactives pour savoir s'il y a eu un problème de bourrage à l'impression des petites listes et/ou des tableaux.
- Si les petites listes ont eu un problème il faut récupérer le fichier TRVLST.Hnn et le renommer TRVLST.DAT pour relancer le programme
- S'il y a eu un problème pendant l'impression des tableaux, suivant que le programme de création des tableaux se sera exécuté jusqu'au bout ou pas , il aura ou non changé les valeurs particulières (Cfr 10.5.3)

. S'il les a changées, l'utilisateur introduit les bonnes valeurs et le programme des tableaux peut être relancé.

10.6.4 Utilisation.

- écran 1 : le programme affiche le test de tous les fichiers utilisés
- écran 2 : quelques questions sont posées afin de savoir quel est le problème

- écran 3 : il y a des cas où le programme ne peut rien faire
- écran 4 : le programme demande si certaines valeurs sont exactes puis ...
- écran 5 : signale que l'on peut relancer le programme d'impression des tableaux

écran 1

```
                PROGRAMME DE SAUVETAGE/RECUPERATION.
                ------------------------------------

TEST DE TOUS LES FICHIERS UTILISES

   TEST FICHIER 'B:FILDDE.DAT ' : OK
   TEST FICHIER 'B:FILREF.DAT ' : OK
   TEST FICHIER 'A:FILVAL.DAT ' : OK
   TEST FICHIER 'A:FNOLST.DAT ' : File not found
   TEST FICHIER 'B:NROLST.DAT ' : File not found
   TEST FICHIER 'B:NROSMA.DAT ' : File not found
   TEST FICHIER 'B:PLATEAU.DAT' : File not found
   TEST FICHIER 'A:FANAL.DAT  ' : OK
   TEST FICHIER 'B:LASTREF.DAT' : File not found
   TEST FICHIER 'B:TRVLST.DAT ' : OK
```

écran 2

```
                PROGRAMME DE SAUVETAGE/RECUPERATION.
                ------------------------------------

Les listes sont-elles sorties correctement (o/n) ? : Non
Y a-t-il eu un bourrage lors de l'impression (o/n) ? : Oui
L'impression des tableaux s'est elle bien deroulee (o/n) ? : Oui
L'impression des petites listes s'est elle bien deroulee (o/n) ? : Non
```

écran 3

```
DESOLE !!! je ne peux pas remedier a la situation.

Adressez-vous a un informaticien ...
```

écran 4

```
                    DIAGNOSTIC.
                    -----------

La derniere reference traitee en tableaux est Z66660
Est-ce correct (o/n) ? : Non
Donnez la derniere reference traitee puis (return) : N45698
Les listes precedentes portaient le numero 0
Est-ce correct (o/n) ? : Non
Donnez le numero des dernieres listes : 2
La derniere reference en SMA12 portait le numero 56
Est-ce correct (o/n) ? : Non
Donnez le numero de la derniere reference en SMA12 : 27
Le dernier plateau cobas portait le numero 5
Est-ce correct (o/n) ? : Non
Donnez le numero du dernier plateau cobas : 3
```

écran 5

Vous pouvez relancer le programme TABLEAUX !

10.6.5 Texte source.

```
/* PROGRAMME DE RECUPERATION EN CAS DE BOURRAGE A L'IMPRESSION */

#include "A:BDSCIO.H"
#include "a:screen.c";
#define dim_liste 500

char iobuf[BUFSIZ],last[7];

int nrolst,lastref,filref,num;
int fildde,liste_ok,tableaux_ok,ptliste_ok,encodage;
int ligne,filval,fnolst,nrosma,plateau,fanal,it,trvlst;
int bourrage,nb_ref;
int heure;
char zone[3];
char newname[14];

struct nro_ref { char nro[7];};

struct { struct nro_ref reff;
         int ndde;} liste[dim_liste];


/* CLOTURE DU PROGRAMME */
int fini()
(for (it=1;it<=3;it++) bell
home;
cteos;
printf("OK");
exit();
}
```

```
main()
{cadre();ligne = 2;
pos(ligne++,18); printf("PROGRAMME DE SAUVETAGE/RECUPERATION.");
pos(ligne++,18); printf("-----------------------------------");
ligne++;
pos(ligne++,2); printf("TEST DE TOUS LES FICHIERS UTILISES");
ligne++;
pos(ligne++,5); printf("TEST FICHIER 'B:FILDDE.DAT ' : ");
fildde = open ("b:fildde.dat",0);
if (fildde == -1) printf("%s",errmsg(errno())); else printf("OK");
close (fildde);
pos(ligne++,5); printf("TEST FICHIER 'B:FILREF.DAT ' : ");
filref = open ("b:filref.dat",0);
if (filref == -1) printf("%s",errmsg(errno())); else printf("OK");
close (filref);
pos(ligne++,5); printf("TEST FICHIER 'A:FILVAL.DAT ' : ");
filval = open ("a:filval.dat",0);
if (filval == -1) printf("%s",errmsg(errno())); else printf("OK");
close (filval);
pos(ligne++,5); printf("TEST FICHIER 'A:FNOLST.DAT ' : ");
fnolst = open ("a:fnolst.dat",0);
if (fnolst == -1) printf("%s",errmsg(errno())); else printf("OK");
close (fnolst);
pos(ligne++,5); printf("TEST FICHIER 'B:NROLST.DAT ' : ");
nrolst = open ("b:nrolst.dat",0);
if (nrolst == -1) printf("%s",errmsg(errno())); else printf("OK");
close (nrolst);
pos(ligne++,5); printf("TEST FICHIER 'B:NROSMA.DAT ' : ");
nrosma = open ("b:nrosma.dat",0);
if (nrosma == -1) printf("%s",errmsg(errno())); else printf("OK");
close (nrosma);
pos(ligne++,5); printf("TEST FICHIER 'B:PLATEAU.DAT' : ");
plateau = open ("b:plateau.dat",0);
if (plateau == -1) printf("%s",errmsg(errno())); else printf("OK");
close (plateau);
pos(ligne++,5); printf("TEST FICHIER 'A:FANAL.DAT  ' : ");
fanal = open ("a:fanal.dat",0);
if (fanal == -1) printf("%s",errmsg(errno())); else printf("OK");
close (fanal);
pos(ligne++,5); printf("TEST FICHIER 'B:LASTREF.DAT' : ");
lastref = open ("b:lastref.dat",0);
if (lastref == -1) printf("%s",errmsg(errno())); else printf("OK");
close (lastref);
pos(ligne++,5); printf("TEST FICHIER 'B:TRVLST.DAT ' : ");
trvlst = open ("b:trvlst.dat",0);
if (trvlst == -1) printf("%s",errmsg(errno())); else printf("OK");
close (trvlst);
temporisation();
```

```
cadre(); ligne = 2;
pos(ligne++,18); printf("PROGRAMME DE SAUVETAGE/RECUPERATION.");
pos(ligne++,18); printf("-----------------------------------");
ligne++;
liste_ok = oui(ligne++,2,"Les listes sont-elles sorties correctement",'
;

if (! liste_ok)
  (bourrage = oui(ligne++,2,
        "Y a-t-il eu un bourrage lors de l'impression",'n');
   tableaux_ok = oui(ligne++,2,
        "L'impression des tableaux s'est elle bien deroulee",'n');
   ptliste_ok = oui(ligne++,2,
        "L'impression des petites listes s'est elle bien deroulee",'n').
  )
  else (bourrage = 0; tableaux_ok = 1;  ptliste_ok = 1; )
temporisation();
cadre(); ligne = 2;
pos(ligne++,33); printf("DIAGNOSTIC.");
pos(ligne++,33); printf("-----------");
ligne++;
if (! tableaux_ok)
(fopen("b:lastref.dat",iobuf);
  fscanf(iobuf,"%s\n",last);
  pos(ligne++,3);
  printf("La derniere reference traitee en tableaux est %s",last);
  if (! oui(ligne++,3,"Est-ce correct"))
  (pos(ligne++,3);
   printf("Donnez la derniere reference traitee puis <return> : ");
   scanf("%s\n",last);
   fclose(iobuf);
   fcreat("b:lastref.dat",iobuf);
   fprintf(iobuf,"%s\n",last);
   )
  fclose(iobuf);
  fopen("b:nrolst.dat",iobuf);
  fscanf(iobuf,"%d\n",&num);
  pos(ligne++,3);
  printf("Les listes precedentes portaient le numero %d",num);
  if (! oui(ligne++,3,"Est-ce correct"))
  (pos(ligne++,3);
   printf("Donnez le numero des dernieres listes : ");
   scanf("%d\n",&num);
   fclose(iobuf);
   fcreat("b:nrolst.dat",iobuf);
   fprintf(iobuf,"%d\n",num);
   )
   fclose(iobuf);
```

```
    fopen("b:nrosma.dat",iobuf);
    fscanf(iobuf,"%d\n",&num);
    pos(ligne++,3);
    printf("La derniere reference en SMA12 portait le numero %d",num);
    if (! oui(ligne++,3,"Est-ce correct"))
    {pos(ligne++,3);
     printf("Donnez le numero de la derniere reference en SMA12 : ");
     scanf("%d\n",&num);
     fclose(iobuf);
     fcreat("b:nrosma.dat",iobuf);
     fprintf(iobuf,"%d\n",num);
     }
     fclose(iobuf);


    fopen("b:plateau.dat",iobuf);
    fscanf(iobuf,"%d\n",&num);
    pos(ligne++,3);
    printf("Le dernier plateau cobas portait le numero %d",num);
    if (! oui(ligne++,3,"Est-ce correct"))
    {pos(ligne++,3);
     printf("Donnez le numero du dernier plateau cobas : ");
     scanf("%d\n",&num);
     fclose(iobuf);
     fcreat("b:plateau.dat",iobuf);
     fprintf(iobuf,"%d\n",num);
     }
     fclose(iobuf);
     cadre();
     pos(10,3);
     printf("Vous pouvez relancer le programme TABLEAUX !");
     temporisation();
     fini();
}
```

```
if (! ptliste_ok)
(if (open("b:trvlst.dat",0) == -1)
 if (errno() == 11)
 (fclose(trvlst);
  heure = peek(call(61497,0,0,0,0)+3);
  sprintf(zone,"%d",heure);
  strcpy(newname,"b:trvlst.h");
  strcat(newname,zone);
  rename(newname,"b:trvlst.dat");
  cadre();
  pos(10,3);
  printf("Vous pouvez relancer le programme PTLISTE !");
  temporisation();
  fini();
 )
)
cadre();
pos(10,3);
printf("DESOLE !!! je ne peux pas remedier a la situation.");
pos(12,3);
printf("Adressez-vous a un informaticien ...");
temporisation();
fini();
}
```

10.7 Le programme de récupération en cas de panne.

10.7.1 Signification des principales variables.

liste — table contenant les références du fichier FILREF

10.7.2 Description des fonctions.

int charger_ref()
: chargement de toutes les références de FILREF dans liste.

int trt_trvlst()
: essaye de récupérer la dernière version sauvée de TRVLST.

int ref_pg_sect()
: cette routine est appelée quand on trouve plus de références que de tableaux de demandes. S'il y a 7 références de plus que de demandes elle détruit les 7 dernières références pour restaurer la bijection entre les fichiers FILREF et FILDDE. (Cfr appendice I du premier volume)

int sect_pg_ref()
: cette routine est utilisée dans le cas inverse (plus de tableaux de demandes que de références) elle détruit alors les n dernières demandes pour les mêmes raisons que celles évoquées plus haut.

10.7.3 Programme principal.

- charger et compter les références
- compter les demandes
- si ces deux nombres ne sont pas égaux, effectuer le traitement approprié (Cfr fonctions)
- essayer de récupérer le fichier TRVLST (lignes pour les petites listes)

10.7.4 Utilisation.

Il y a trois cas possibles:

- écran 1 : tout est normal
- écran 2 : plus de références que de demandes
- écran 3 : plus de demandes que de références

écran 1

```
RECUPERATION DES FICHIERS APRES UNE PANNE DE COURANT.
-----------------------------------------------------

        2 REFERENCES.
        2 DEMANDES.
        Tout semble ok..... mais il se peut que les dernieres
        references (MAXIMUM 16) encodees aient ete perdues .
        Derniere reference enregistree : Z66662 .

        le fichier contenant les listes de travail a ete detruit
        on va essayer de recuperer le dernier backup
        La reference J50019 (analyse 113)  est-elle deja
        passee en petites listes (o/n) ? :
```

écran 2

```
RECUPERATION DES FICHIERS APRES UNE PANNE DE COURANT.
-----------------------------------------------------

        3 REFERENCES.
        2 DEMANDES.
        on va retirer 1 reference  !

        Normalement, tout est arrange.
        La correspondance entre les fichiers est retablie.
        Derniere reference enregistree : Z66662 .

        Le fichier des listes de travail n'a subit
        aucun dommage.
```

écran 3

```
RECUPERATION DES FICHIERS APRES UNE PANNE DE COURANT.
-----------------------------------------------------

        1 REFERENCE .
        2 DEMANDES.

        Normalement, tout est arrange.
        La correspondance entre les fichiers est retablie.
        Derniere reference enregistree : Z66663 .

        Le fichier des listes de travail n'a subit
        aucun dommage.
```

10.7.5 Texte source.

```
/* PROGRAMME DE RECUPERATION EN CAS DE PANNE DE COURANT */

#include "a:bdscio.h"
#include "a:screen.c"
#define dim_liste 500

struct st_buf   {char lettre;
                 char ref[5];
                 char car_return  ;
                 char line_feed   ;
                }out_ref_buffer[16],*ptr_buf;

char filref[BUFSIZ],trvlst[BUFSIZ],trvbck[BUFSIZ],tmps[13],ptmps[13];

int dde[64],fildde,nb_ref,nb_sect,ligne,i,filbck;

struct nro_ref { char nro[7];};

struct { struct nro_ref reff;
         int ndde;} liste[dim_liste];



/* CHARGEMENT DES REFERENCES DE FILREF */
int charger_ref()
{int i,sect;
 struct nro_ref rf;
 i = -1;
 sect = -1;
  while ((fscanf(filref,"%s\n",rf) > 0) && (i < dim_liste))
        {strcpy(liste[++i].reff,rf);
         liste[i].ndde = ++sect;
         if (! isupper(rf.nro[0])) {i--; sect--;}
        }
  nb_ref = i + 1;
}
```

```
/* RECUPERATION DU FICHIER  TRVLST */
int trt_trvlst()
{int i,j,n,k;
struct nro_ref ref;
i=0; ligne++;
fopen("b:trvlst.dat",trvlst);
while(fgets(tmps,trvlst) != 0) i++;
fclose(trvlst);
if (i == 0)
    {pos (ligne++,20);
     printf("le fichier contenant les listes de travail a ete detruit")
     pos (ligne++,20);
     printf("on va essayer de recuperer le dernier backup ");
     unlink ("b:trvlst.dat");
     if (fopen("b:trvlst.bck",trvbck) != -1)
     {k =0;
      while(fgets(tmps,trvbck) != 0) {strcpy(ptmps,tmps);  k++;}
      fclose(trvbck);
      if ((k>0) && (sscanf(ptmps,"%d%s\n",&n,ref) > 0) )
      {pos(ligne++,20);
       printf("La reference %s (analyse %d)  est-elle deja",ref,n);
       if (! oui(ligne++,20,"passee en petites listes",'n'))
            rename("b:trvlst.bck","b:trvlst.dat");
      }
     }else
      {pos(ligne++,20); printf("Desole je ne trouve pas de backup !");
       pos(ligne++,20);
       printf("Si vous n'avez pas encode plus de 16 references");
       pos(ligne++,20);
       printf("depuis les dernieres listes, c'est normal .");
      }
    }else  {pos(ligne++,20);
        printf("Le fichier des listes de travail n'a subit");
        pos(ligne++,20);  printf("aucun dommage.");
           }
}
```

```
/* TRAITEMENT QUAND IL Y A PLUS DE REFERENCES QUE DE SECTEURS */
int ref_pg_sect()
{int filref,nn,entrop,last;
entrop = nb_ref - nb_sect;
last = nb_ref - 1 - entrop;
pos (ligne++,20);
printf("on va retirer %d references !",entrop);
filref = open("b:filref.dat",2);
seek (filref,-1,2);
read (filref,out_ref_buffer,1);
ptr_buf = out_ref_buffer;
for (nn = 0;isupper((ptr_buf++)->lettre);nn++);
ptr_buf = out_ref_buffer[nn-1];
for (;entrop >0; entrop--) setmem(ptr_buf--,8,'\0');
seek (filref,-1,2);
write(filref,out_ref_buffer,1);
close (filref);
ligne++;
pos (ligne++,20);
printf("Normalement, tout est arrange.");
pos (ligne++,20);
printf("La correspondance entre les fichiers est retablie.");
pos (ligne++,20);
printf("Derniere reference enregistree : %s .",liste[last]);
}

/* TRAITEMENT QUAND IL Y A PLUS DE SECTEURS QUE DE REFERENCES */
int sect_pg_ref()
{rename("b:fildde.dat","b:fildde.bck");
fildde = creat("b:fildde.dat");
filbck = open("b:fildde.dat",0);
for (i=1;i<=nb_ref;i++)
        { read (filbck,dde,1);
          write(fildde,dde,1);
        }
close(fildde);
close(filbck);
unlink("b:filbck.dat");
ligne++;
pos (ligne++,20);
printf("Normalement, tout est arrange.");
pos (ligne++,20);
printf("La correspondance entre les fichiers est retablie.");
pos (ligne++,20);
printf("Derniere reference enregistree : %s .",liste[nb_ref-1]);
}
```

```
main()
{
cadre();
ligne = 2;
pos (ligne++,10);
printf("RECUPERATION DES FICHIERS APRES UNE PANNE DE COURANT.");
pos (ligne++,10);
printf("-----------------------------------------------------");
fopen("b:filref.dat",filref);
fildde = open("b:fildde.dat",0);
charger_ref();
ligne++;
pos (ligne++,20);
printf("%d REFERENCES.",nb_ref);
nb_sect = cfsize(fildde);
pos (ligne++,20);
printf("%d DEMANDES.",nb_sect);
close (fildde);
fclose (filref);
if (nb_sect == nb_ref)
        (pos (ligne++,20);
         printf("Tout semble ok..... mais il se peut que les dernieres"
         pos (ligne++,20);
         printf("references (MAXIMUM 16) encodees aient ete perdues .")
         pos (ligne++,20);
         printf("Derniere reference enregistree : %s .",liste[nb_ref-1]
        )
else    if (nb_ref > nb_sect) ref_pg_sect(); else sect_pg_ref();
trt_trvlst();
temporisation();
home;
cteos;
printf("OK");
}
```

Chapter 11

Programme de gestion des analyses.

11.1 Routines de gestion d'ecran.

11.1.1 Signification des principales variables.

```
t_zone          = record
                    typ : integer;  type de la variable lue
                    l   : integer;  ligne a laquelle doit aparaître txt
                    c   : integer;  colonne a laquelle doit aparaître tx
                    txt : string[max_txt_zone]; texte
                  end;
t_map           = array [0..max_map_size] of t_zone;
```

une map est définie comme un ensemble de zones définies comme ci-dessus.

11.1.2 Description des fonctions.

```
PROCEDURE g_erreur (n:INTEGER);
```
Imprime un message d'erreur

```
PROCEDURE cvint (str:str6; var n:integer);
```
renvoie dans n, le string représentant un nombre contenu dans str; renvoie -1 si str ne contenait pas un nombre en lettre.

```
PROCEDURE ini_map (map: t_map; code : integer);
```
Initialise l'écran avec les différentes zones de la map.

```
FUNCTION get_fl :char;
```
renvoie un code suivant le caractère tapé au clavier

```
PROCEDURE load_map (filename: t_filename; var map: t_map);
```
Charge dans map les valeurs du fichier filename

```
PROCEDURE read_map (map:t_map; var nro_zone_cour: integer);
                lit une valeur  pour  la zone nro_zone_cour et la
                renvoie dans une variable globale

PROCEDURE write_map (map: t_map; nro_zone: integer);
                imprime à l'écran une valeur globale dans la zone
                nro_zone
```

11.1.3 Texte source.

```
MODULE gesmap;

CONST
        max_map_size    =25;
        max_txt_size    =30;

TYPE
        str6            = string[6];
        tab27           = array [1..27] of integer;
        t_filename      = string[14];
        t_zone          = record
                                typ : integer;
                                l   : integer;
                                c   : integer;
                                txt : string[max_txt_zone];
                          end;
        t_map           = array [0..max_map_size] of t_zone;

VAR
        l_cour,c_cour   : integer;
        gl_integer      : external integer;
        gl_str10        : external string[10];
        gl_str70        : external string[70];
        gl_str25        : external string[25];
        gl_car          : external char;
        gl_tab_27       : external tab_27;
        str_entier      : string[6];
        mois,annee,jour : external integer;
        last_op         : external integer;
```

```
(MODULE SCREEN)

EXTERNAL FUNCTION  OUI  (ST:STRING):BOOLEAN;
EXTERNAL PROCEDURE CADRE ;
EXTERNAL PROCEDURE UP;
EXTERNAL PROCEDURE RIGHT;
EXTERNAL PROCEDURE DOWN;
EXTERNAL PROCEDURE LEFT;
EXTERNAL PROCEDURE CR;
EXTERNAL PROCEDURE CTEOL;
EXTERNAL PROCEDURE CTEOS;
EXTERNAL PROCEDURE CSCR;
EXTERNAL PROCEDURE HOME;
EXTERNAL PROCEDURE RESTORE;
EXTERNAL PROCEDURE DSPCH        (N:INTEGER);
EXTERNAL PROCEDURE ESC;
EXTERNAL PROCEDURE BELL         (N:INTEGER);
EXTERNAL PROCEDURE DESABLE;
EXTERNAL PROCEDURE ENABLE;
EXTERNAL PROCEDURE BLINK;
EXTERNAL PROCEDURE GRAPHIC;
EXTERNAL PROCEDURE REVERSE;
EXTERNAL PROCEDURE LOWINT;
EXTERNAL PROCEDURE INSERT;
EXTERNAL PROCEDURE POS  (L,C:INTEGER);

EXTERNAL FUNCTION power (nb,puiss:integer):integer;
```

```
PROCEDURE g_erreur (n:INTEGER);

var     i:integer;

begin
        pos(23,30);
        bell(3);
        case n of
        1 : write ('Entrez un nombre entre 0 et 32768 !!!!!');
        else;
        end;
        for i:=1 to maxint do;  for i:=1 to maxint do;
        pos(23,30); cteol;      pos(lcour,c_cour);
end;

PROCEDURE cvint (str:str6; var n:integer);

var     i,lg : integer;

begin
        n:=0;
        lg := length(str);
        if (lg = 0) or (lg >5)
        then n := -1
        else for i := 1 to lg do
                if (str[i] >= '0') and (str[i] <= '9')
                then n:= n*10 + ord(str[i]) - 48
                else begin n := -1; exit; end;
end;

PROCEDURE ini_map (map: t_map; code : integer);

var i : integer;

begin
        case code of
        1 : reverse;
        2 : lowint;
        else;
        end;
        for i:= 1 to map[0].typ do
        begin
                pos(map[i].l,map[i].c);
                write (map[i].txt);
        end;
        home;
        if code <> 0 then desable;
end;
```

```
FUNCTION get_fl :char;

var c: char;

begin
        repeat
        read(c);
        case c of
(^A)    '': get_fl := 'h';
(^B)    '': get_fl := 'b';
(^D)    '': get_fl := 'g';
(^C)    '': get_fl := 'd';
(esc)   '': get_fl := 'e';
        '?' : begin
                get_fl := 'i';
                pos(l_cour,c_cour);
                write (' ');
                pos(l_cour,c_cour);
              end
        else
        if (c>=' ') and (c<='~')
        then begin
                pos(l_cour,c_cour);
                write(' ');
                pos(l_cour,c_cour);
             end;
        end;
        until ((c>='') and (c<='')) or (c='') or (c='?');
end;


PROCEDURE load_map (filename: t_filename; var map: t_map);

var     file_map : text;
        i        : integer;

begin
        assign(file_map,filename);
        reset(file_map);
        i := 0;
        while (not eof (file_map)) and (i<map_max_size)
        do begin
                i := i+1;
                readln(file_map,map[i].typ,map[i].l,
                        map[i].c,map[i].txt);
           end;
        map[0].typ := i;
end;
```

```
PROCEDURE read_map (map:t_map; var nro_zone_cour: integer);

LABEL 7,11,22,88,99;

VAR     i: integer;
        str_entier :string[6];
        c :char;

PROCEDURE posit;
begin
        l_cour := map[nro_zone_cour].l;
        c_cour := map[nro_zone_cour].c;
        pos(l_cour,c_cour);
end;

begin
        if nro_zone_cour < 0
        then begin
                nro_zone_cour := -nro_zone_cour;
             7: nro_zone_cour := nro_zone_cour + 1;
                if nro_zone_cour > map[0].typ then nro_zone_cour := 1;
                if map[nro_zone_cour].typ >= 9 then goto 7;
                posit;
             end;
        if nro_zone_cour = 0
        then begin
                i := 1;
                while (i< map[0].typ) and (map[i].typ >= 9) do i := i+1;
                nro_zone_cour := i;
             end;
        posit;
        99:case get_fl of
        'i': begin nro_zone_cour := nro_zone_cour+1000;exit; end;
        'e': begin nro_zone_cour := -1; exit; end;
        'b': begin
           11: nro_zone_cour :=nro_zone_cour +1;
               if nro_zone_cour > map[0].typ then nro_zone_cour := 1;
               if map[nro_zone_cour].typ >= 9 then goto 11;
               posit;
               goto 99;
             end;
        'h' : begin
             22:nro_zone_cour := nro_zone_cour -1;
                if nro_zone_cour <1 then nro_zone_cour := map[0].typ;
                if map[nro_zone_cour].typ>= 9 then goto 22;
                posit;
                goto 99;
              end;
```

```
'd': begin
     88:c_cour := c_cour + length(map[nro_zone_cour].txt) +1;
        pos(l_cour,c_cour);
        case map[nro_zone_cour].typ of
           0:;
           1: repeat
                read(c);
                if (c='') or (c='')  {^C ou ^D}
                then begin
                     posit;
                     goto 99;
                     end
                     else
                     begin
                     readln(str_entier);
                     cvint(concat(c,str_entier),gl_integer);
                     if gl_integer <0 then g_erreur (1);
                     end;
              until gl_integer >=0;
              2: begin
                  read(c);
                  if (c='') or (c='')  {^C ou ^D}
                  then begin
                      posit;
                      goto 99;
                      end
                      else
                      begin
                      readln((gl_str10);
                      gl_str10 := concat(c,gl_str10);
                      end;
                  end;

              3: begin
                  read(c);
                  if (c='') or (c='')  {^C ou ^D}
                  then begin
                      posit;
                      goto 99;
                      end
                      else
                      begin
                      readln((gl_str70);
                      gl_str70 := concat(c,gl_str70);
                      end;
                  end;
```

```
                    4: begin
                         read(c);
                         if (c='') or (c='')  (^C ou ^D)
                         then begin
                             posit;
                             goto 99;
                             end
                             else
                             begin
                             readln((gl_str25);
                             gl_str25 := concat(c,gl_str25);
                             end;
                         end;

                     5,6: begin
                         read(c);
                         if (c='') or (c='')  (^C ou ^D)
                         then begin
                             posit;
                             goto 99;
                             end
                             else gl_car := c;
                         end;

                     8: begin
                         if (last_op>0) and last_op<max_map_size)
                         then begin
                                 nro_zone_cour := last_op;
                                 l_cour := map[nro_zone_cour].l;
                                 c_cour := map[nro_zone_cour].c;
                                 goto 88;
                              end;
                         else begin
                                 posit;
                                 goto 99;
                               end
                         end
                 else;
                 end;
        end;
        'g' : goto 99;
        else;
        end;
        last_op := nro_zone_cour;
end;
```

```
PROCEDURE write_map (map: t_map; nro_zone: integer);

var i : integer;

begin
        l_cour := map[nro_zone].l;
        c_cour := map[nro_zone].c + length(map[nro_zone].txt) +1;
        pos(l_cour,c_cour);
        case map[nro_zone].typ of
        0:;
        1,11: if gl_integer <> maxint then write (gl_integer:5);
        2: write(gl_str10);
        3: write(gl_str70);
        4: write(gl_str25);
        5: write(gl_car);
        6: write(gl_str25);
        7:;
        8:;
        9:;
        10:write(jour:2,'/',mois:2,'/',annee:2);
        12:for i:=2 to gl_tab27[1] do write(gl_tab27[i]:4);
        else;
        end;

end;

modend.
```

11.2 Gestion des analyses.

11.2.1 Signification des principales variables.

```
t_analyse =
        RECORD
                nro_a           : INTEGER;    numéro de l'analyse
                libelle         : STRING[25]; libellé de l'analyse
                nro_descr       : INTEGER;    numéro de la description d
                                                de cette analyse
        END;

t_description =
        RECORD
                nro_d           : INTEGER;    numéro de description
                prelevement     : CHAR  ;     code de prélèvement
                technique       : CHAR  ;     code technique
                dispatching     : CHAR  ;     code dispatching
                dbl_rincage     : BOOLEAN;    code double rincage
                qte_utilisee    : INTEGER;    quantité de prélèv. utilis
                nro1_tampon     : INTEGER;    numéro du tampon 1
                qte1_tampon     : INTEGER;    quantité de tampon 1
                nro2_tampon     : INTEGER;    numéro du tampon 2
                qte2_tampon     : INTEGER;    quantité de tampon 2
        END;

t_commentaire =
        RECORD
                date            : INTEGER       ; date de création
                txt             : STRING[70]    ; commentaire
                list_analyse    : tab27         ; liste des numéros des
                                                  analyses décrites par
                                                  cette description.
        END;
```

11.2.2 Description des fonctions.

```
FUNCTION valid_descr (n_anal,n_descr : INTEGER): BOOLEAN;
                si la description  n_descr  existe,  la  fonction
                ajoute n_anal dans le champ list_analyse de cette
                description sinon, renvoie false.

PROCEDURE temporisation;
                stoppe le programme tant  que  l'utilisateur  n'a
```

```
                pas pressé une touche.

PROCEDURE cv_entier_en_date (d: INTEGER; VAR j,m,a: INTEGER);
                convertit l'entier d en date : j,m,a

PROCEDURE cv_date_en_entier (VAR d: INTEGER; j,m,a: INTEGER);
                convertit les entiers j,m,a en un seul entier d;

PROCEDURE next_page(VAR nro : INTEGER;j,m,a: INTEGER) ;
                effectue un saut de page et initialise la
                nouvelle page avec le numéro de page et la date.

PROCEDURE entete;
                Imprime une page d'entête au nom du laboratoire.

PROCEDURE stopper;
                ferme les fichiers et demande confirmation avant
                de sortir du programme

PROCEDURE erreur(n: INTEGER);
                affiche le message d'erreur correspondant au code
                n

PROCEDURE MENU; affiche le menu principal et renvoie le choix de
                l'utilisateur

PROCEDURE anal_ajouter;
                gère toutes les opérations nécessaires à l'ajoute
                d'une analyse

PROCEDURE anal_detruire;
                détruit une analyse

PROCEDURE anal_consulter;
                affiche à l'écran la description d'une analyse

PROCEDURE anal_modifier;
                effectue les opérations permettant de modifier
                une analyse

PROCEDURE desc_ajouter;
                gère toutes les opérations nécessaires à l'ajoute
                d'une description

PROCEDURE  ini;  initialise avec des valeurs par défaut les
                différents champs d'une description

PROCEDURE wr_prelevement (ecran: BOOLEAN);
                affiche ou imprime le type de prélèvement
```

```
PROCEDURE wr_dispatching (ecran: BOOLEAN);
              affiche ou imprime le type de dispatching

PROCEDURE wr_technique (ecran: BOOLEAN);
              affiche ou imprime le type de technique

PROCEDURE wr_dbl_rincage (ecran: BOOLEAN);
              affiche  ou imprime "oui" ou "non" suivant  qu'il
              faut un double rincage ou pas

PROCEDURE wr_list_analyse;
              affiche les valeurs du champ list_analyse

PROCEDURE wr_integer(n: INTEGER);
              affiche un nombre entier

PROCEDURE help ( n : INTEGER);
              affiche le message help correspondant au code n

PROCEDURE desc_detruire;
              détruit une description

PROCEDURE desc_modifier;
              effectue les  opérations  permettant  de modifier
              une description

PROCEDURE desc_consulter;
              affiche les différentes valeurs d'une description

PROCEDURE liste1;
              fournit un listing général des descriptions

PROCEDURE liste2;
              fournit   un   listing   avec   le   détail   des
              descriptions

PROCEDURE liste3;
              fournit un listing des analyses
```

11.2.3 Programme principal.

- initialisations
- introduction de la date
- introduction du mode d'écran (normal, inverse, basse intensité)
- répéter: afficher menu et effectuer l'opération choisie

11.2.4 Utilisation.

- Le programme commence par demander la date (écran 1)

- Il demande ensuite le mode d'ecran (écran 2): normal (blanc sur noir), reverse (noir sur blanc), et faible intensité. Ceci signifie que si par exemple l'utilisateur choisit "faible intensité", les noms des zones sur un écran apparaîtront moins lumineux que le contenu introduit pour ces zones.

- Le programme demande si on veut travailler sur de nouveaux fichiers, sinon on garde les fichiers existants.(écran 3).

- Il affiche ensuite le menu principal.(écran 4). Le curseur se trouve devant "Replay". Un choix menu s'effectue de la manière suivante: on utillise la flèche vers le haut ou la flèche vers le bas pour descendre ou monter dans le menu et quand le curseur se trouve devant la ligne choisie, on tape sur la flèche vers la droite; dans les 8 premiers cas, il faut introduire un numéro, dans les 4 derniers le traitement s'enclanche directement.

- Nous voulons ajouter l'analyse 14 (écran 5)

- L'écran affiché laisse 2 zones à remplir (écran 6): le libellé et le numéro de description. Le système est le même que pour le menu principal: les flèches vers le haut ou vers le bas permettent de passer à la zone précédente ou suivante et la flèche vers la droite permet d'entrer dans la zone pour y introduire une valeur.

- Donnons donc le deux valeurs requises et nous obtenons l'écran 7. On tape <return> pour enregistrer et le programme affiche à nouveau le menu. Choisissons de consulter l'analyse 14 et nous obtenons l'écran 8. La description 4 existait déjà dans les fichiers et le programme nous demande si nous voulons visionner cette description. Répondons oui et nous obtenons l'écran 9.

- Apres avoir consulté cet écran, on tape sur une touche et on se retrouve au menu. Choisissons de modifier l'analyse 14. Les manipulations avec les flèches sont les mêmes et l'on peut ainsi corriger les erreurs.(écran 10)

- Consultons à nouveau l'analyse 14 et on s'appercoit que les corrections ont bien été enregistrées.(écran 11)

- Revenons au menu et détruisons l'analyse 14.(écran 12)

- Une consultation de cette analyse (écran 13) affichera un message d'erreur.

- Choisissons d'ajouter la description 18. Le programme affiche un écran avec des zones à remplir.(écran 14) On peut toujours voyager dans l'écran avec les flèches. Les zones PRELEVEMENT, DISPATCHING et TECHNIQUE se remplissent en pressant une lettre code, les autres en introduisant une valeur suivie de <return>

- Une fois l'écran rempli, nous obtenons l'écran 15

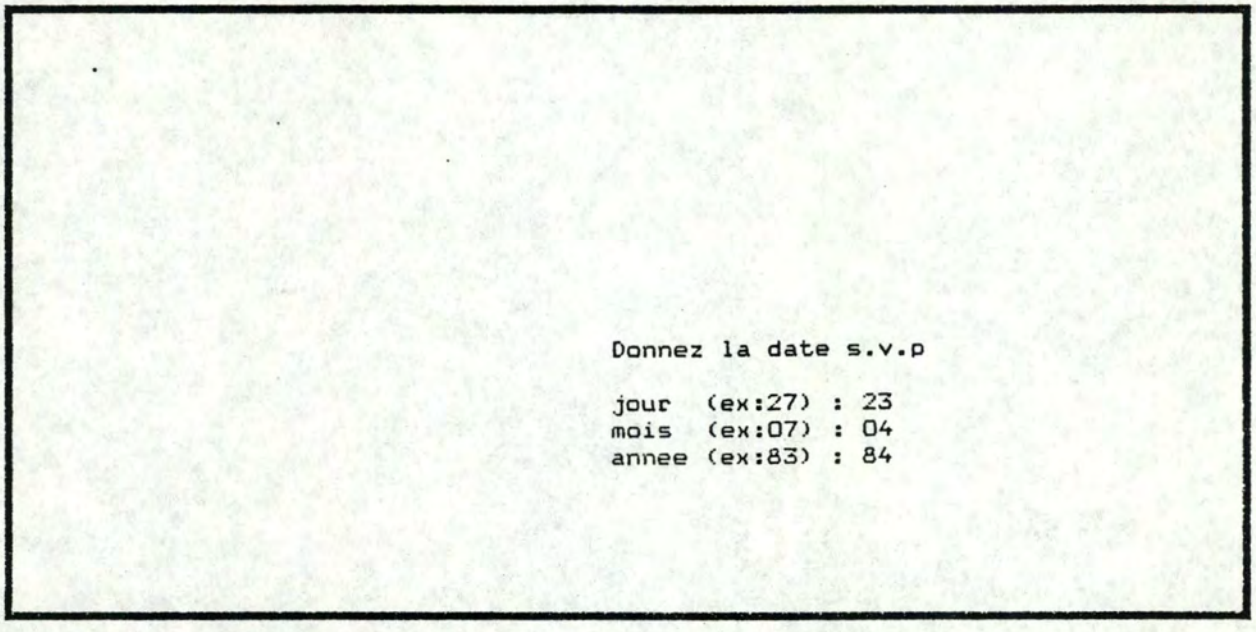
- Ajoutons la description 19. Si on ne connait plus les codes, on peut taper un "?" au moment ou le curseur est devant une certaine zone et obtenir ainsi de l'information sur ce qu'il faut mettre dans cette zone. L'écran 16 a été obtenu en tapant un "?" lorsque le curseur se trouvait devant PRELEVEMENT, l'écran 17 avec le curseur devant QUANTITE UTILISEE, l'écran 18 avec le curseur devant TECHNIQUE, et enfin l'écran 19 avec le curseur devant REM.

- L'écran 20 est le résultat d'une consultation de la description 18

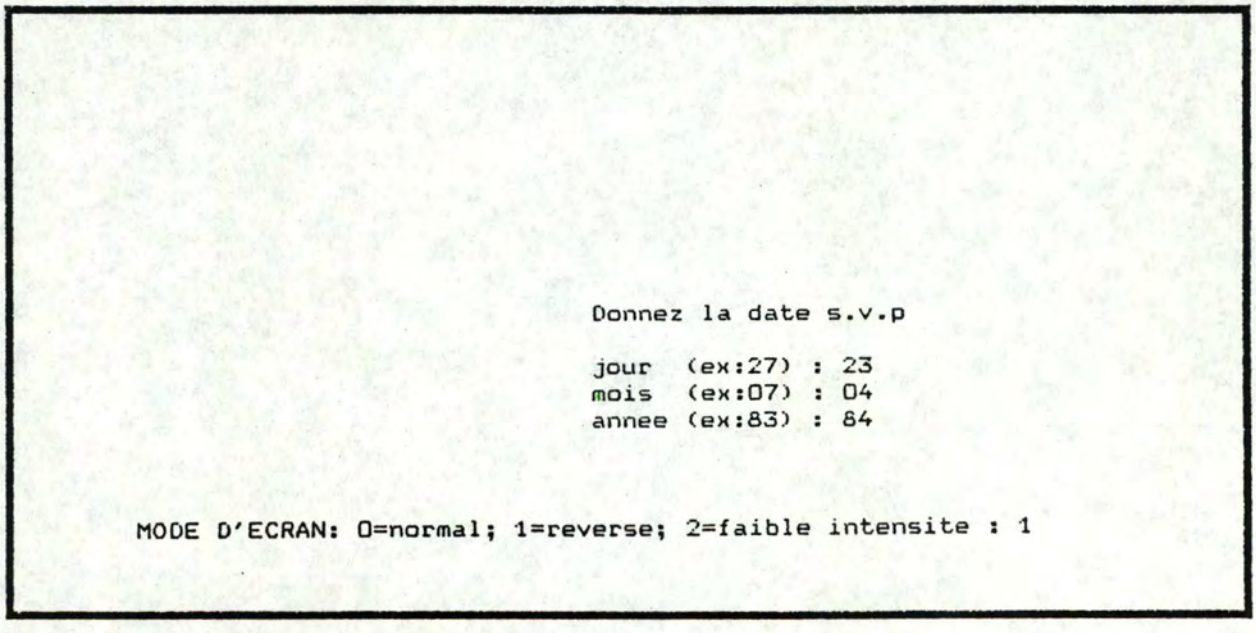
- Choisissons au menu LISTE DES DESCRIPTIONS et nous obtenons l'écran 21. Il suffira par après d'imprimer le fichier LISTE1.LPT.

- Choisissons enfin de stopper le programme. L'écran 22 demande confirmation de l'arrêt du programme avant de stopper définitivement.

écran 1



écran 2



écran 3

```
CREATION DE 3 NOUVEAUX FICHIERS (o/n) ? :
```

écran 4

```
                GESTION DES ANALYSES
                --------------------

Replay
Ajouter    analyse      #
Detruire   analyse      #
Consulter  analyse      #
Modifier   analyse      #

Ajouter    description  #
Detruire   description  #
Consulter  description  #
Modifier   description  #

Liste des descriptions
Liste par description
Liste des analyses

Stopper ce programme
```

écran 5

```
                GESTION DES ANALYSES
                --------------------

Replay
Ajouter    analyse      # 14
Detruire   analyse      #
Consulter  analyse      #
Modifier   analyse      #

Ajouter    description  #
Detruire   description  #
Consulter  description  #
Modifier   description  #

Liste des descriptions
Liste par description
Liste des analyses

Stopper ce programme
```

écran 6

```
ANALYSE #     14     LIBELLE:                              DESCRIPTION #
```

écran 7

```
ANALYSE #     14      LIBELLE: AMINIGLOCOSIDE                DESCRIPTION # 4
```

écran 8

```
ANALYSE #     14      LIBELLE: AMINIGLOCOSIDE                DESCRIPTION #     4

VOULEZ-VOUS LA DESCRIPTION COMPLETE DE CETTE ANALYSE(o/n) ? :
```

écran 9

```
ANALYSE #     14      LIBELLE: AMINIGLOCOSIDE              DESCRIPTION #      4

PRELEVEMENT: DIVERS                        QUANTITE UTILISEE:     45  micro-L.

DISPATCHING: BATCH                         TECHNIQUE         : DIVERS

DBL RINCAGE: NON

TAMPON 1    :      6                       QUANTITE          :   180  micro-L.

TAMPON 2    :                              QUANTITE          :        micro-L.

DATE        :  8/11/83

REM:
ANALYSES    :     8   9  28  31  40   1  16  14
```

Tapez sur une touche pour continuer

écran 10

```
ANALYSE #     14      LIBELLE: AMINOGLUCOSIDE              DESCRIPTION #      1
```

écran 11

```
ANALYSE #    14     LIBELLE: AMINOGLUCOSIDE              DESCRIPTION #     1

PRELEVEMENT: DIVERS                        QUANTITE UTILISEE:    20  micro-L.

DISPATCHING: BATCH                         TECHNIQUE         : DIVERS


DBL RINCAGE: OUI

TAMPON 1   :     5                         QUANTITE          :    50  micro-L.

TAMPON 2   :                               QUANTITE          :        micro-L.

DATE       :  8/11/83

REM:
ANALYSES   :    0   6   7  12  18  22  33  14  14
```

Tapez sur une touche pour continuer

écran 12

```
                    GESTION DES ANALYSES
                    --------------------

          Replay
          Ajouter    analyse      #
          Detruire   analyse      # 14
          Consulter  analyse      #
          Modifier   analyse      #

          Ajouter    description  #
          Detruire   description  #
          Consulter  description  #
          Modifier   description  #

          Liste des descriptions
          Liste par description
          Liste des analyses

          Stopper ce programme
```

écran 13

```
                    GESTION DES ANALYSES
                    --------------------

Replay
Ajouter    analyse       #
Detruire   analyse       #
Consulter  analyse       # 14
Modifier   analyse       #

Ajouter    description   #
Detruire   description   #
Consulter  description   #
Modifier   description   #

Liste des descriptions
Liste par description
Liste des analyses

Stopper ce programme
```

ERREUR: CETTE ANALYSE N'EXISTE PAS !!!

écran 14

```
                                                DESCRIPTION #     18

PRELEVEMENT:                    QUANTITE UTILISEE:          micro-L.

DISPATCHING:                    TECHNIQUE         :

DBL RINCAGE:

TAMPON 1    :                   QUANTITE          :         micro-L.

TAMPON 2    :                   QUANTITE          :         micro-L.

DATE        : 23/ 4/84

REM:
ANALYSES    :
```

écran 15

```
                                                    DESCRIPTION #     18

PRELEVEMENT: L.C.R                    QUANTITE UTILISEE: 125     micro-L.

DISPATCHING: BATCH                    TECHNIQUE        : ISOTOPES

DBL RINCAGE: NON

TAMPON 1    : 2                       QUANTITE         : 125     micro-L.

TAMPON 2    :                         QUANTITE         :         micro-L.

DATE        : 23/ 4/84

REM: CECI EST LA DESCRIPTION 18
ANALYSES    :
```

écran 16

```
                                                    DESCRIPTION #     19

PRELEVEMENT:                          QUANTITE UTILISEE:        micro-L.

DISPATCHING:                          TECHNIQUE        :

DBL RINCAGE:

TAMPON 1    :                         QUANTITE         :        micro-L.

TAMPON 2    :                         QUANTITE         :        micro-L.

DATE        : 23/ 4/84

REM:
ANALYSES    :
-------------------------------------------------------------------------------
LIQ. AMNIOTIQUE = A, SANG      = S, URINE  = U, DIVERS = D
LIQ. GASTRIQUE  = G, PONCTION = P, SELLES = F, L.C.R. = L
```

Tapez sur une touche pour continuer

écran 17

```
                                                    DESCRIPTION #     19
 PRELEVEMENT:                          QUANTITE UTILISEE:          micro-L.
 DISPATCHING:                          TECHNIQUE         :
 DBL RINCAGE:
 TAMPON 1    :                         QUANTITE          :         micro-L.
 TAMPON 2    :                         QUANTITE          :         micro-L.
 DATE        : 23/ 4/84
 REM:
 ANALYSES    :
-------------------------------------------------------------------------------
QUANTITE DE PRELEVEMENT UTILISEE  PAR ANALYSE (en micro-L.)
```

Tapez sur une touche pour continuer

écran 18

```
                                                    DESCRIPTION #     19
 PRELEVEMENT:                          QUANTITE UTILISEE:          micro-L.
 DISPATCHING:                          TECHNIQUE         :
 DBL RINCAGE:
 TAMPON 1    :                         QUANTITE          :         micro-L.
 TAMPON 2    :                         QUANTITE          :         micro-L.
 DATE        : 23/ 4/84
 REM:
 ANALYSES    :
-------------------------------------------------------------------------------
TECHNIQUE UTILISEE ... ISOTOPES = I , DIVERS = D
```

Tapez sur une touche pour continuer

écran 19

```
                                                    DESCRIPTION #     19
PRELEVEMENT:                          QUANTITE UTILISEE:          micro-L.
DISPATCHING:                          TECHNIQUE         :
DBL RINCAGE:
TAMPON 1    :                         QUANTITE          :         micro-L.
TAMPON 2    :                         QUANTITE          :         micro-L.
DATE        : 23/ 4/84
REM:
ANALYSES    :
-------------------------------------------------------------------------
COMMENTAIRE (max 70 carac)
```

Tapez sur une touche pour continuer

écran 20

```
                                                    DESCRIPTION #     18
PRELEVEMENT: L.C.R                    QUANTITE UTILISEE:    125  micro-L.
DISPATCHING: BATCH                    TECHNIQUE         : ISOTOPES
DBL RINCAGE: NON
TAMPON 1    :     2                   QUANTITE          :    125  micro-L.
TAMPON 2    :                         QUANTITE          :         micro-L.
DATE        : 23/ 4/84
REM: CECI EST LA DESCRIPTION 18
ANALYSES    :
```

Tapez sur une touche pour continuer

écran 21

```
                    DESCR #  18


CREATION DU FICHIER "B:LISTE1.LPT"  EN COURS ....

CREATION : [OK]
```

Tapez sur une touche pour continuer

écran 22

```
FICHIERS SAUVES ...

FIN DE PROGRAMME : CONFIRMATION(o/n) ? :
```

11.2.5 Texte source.

```
PROGRAM gestion_des_analyses;

CONST   max_map_size     = 16
        max_txt_size     = 30
        menu_a_a         = 4
        menu_d_a         = 5
        menu_c_a         = 6
        menu_m_a         = 7
        menu_a_d         = 8
        menu_d_d         = 9
        menu_c_d         = 10
        menu_m_d         = 11
        menu_lst1        = 12
        menu_lst2        = 13
        menu_lst3        = 14
        menu_stop        = 15



TYPE

        t_filename       = STRING[14]

        t_zone           = RECORD
                             typ : INTEGER
                             l   : INTEGER
                             c   : INTEGER
                             txt : STRING[max_txt_size]
                           END

        t_map            = ARRAY [0..max_map_size] OF t_zone

        tab27    = ARRAY [1..27] OF INTEGER

        t_analyse =
                RECORD
                        nro_a           : INTEGER
                        libelle         : STRING[25]
                        nro_descr       : INTEGER
                END;
```

```
t_description =
        RECORD
                nro_d              : INTEGER
                prelevement        : CHAR
                technique          : CHAR
                dispatching        : CHAR
                dbl_rincage        : BOOLEAN
                qte_utilisee       : INTEGER
                nro1_tampon        : INTEGER
                qte1_tampon        : INTEGER
                nro2_tampon        : INTEGER
                qte2_tampon        : INTEGER
        END;

t_commentaire =
        RECORD
                date               : INTEGER
                txt                : STRING[70]
                list_analyse       : tab27
        END;
```

```
VAR

        set_technique    : SET OF CHAR
        set_prelevement  : SET OF CHAR
        set_dispatching  : SET OF CHAR

        filcom           : FILE OF t_commentaire
        filana           : FILE OF t_analyse
        fildsc           : FILE OF t_description

        lpt              : TEXT
        err,nro          : INTEGER
        ecran            : BOOLEAN
        fichier          : BOOLEAN
        i,j,k            : INTEGER
        choix            : CHAR
        result           : INTEGER
        tmpstr           : STRING[25]
        date_du_jour     : INTEGER
        jour,mois,annee  : INTEGER
        date_j,date_m    : INTEGER
        date_a           : INTEGER
        anal_map         : t_map
        desc_map         : t_map
        menu_map         : t_map
        menu_z_cour      : INTEGER
        desc_z_cour      : INTEGER
        anal_z_cour      : INTEGER
        gl_INTEGER       : INTEGER
        gl_str10         : STRING[10]
        gl_str25         : STRING[25]
        gl_str70         : STRING[70]
        gl_car           : CHAR
        gl_tab27         : tab27
        screen_mode      : INTEGER
        last_menu        : INTEGER
        last_op          : INTEGER
```

```
(declaration des routines externes
-----------------------------------)
(MODULE SCREEN)
EXTERNAL FUNCTION  INPUTINT     (ST:STRING;c:CHAR):INTEGER;
EXTERNAL FUNCTION  OUI          (ST:STRING;c:CHAR):BOOLEAN;
EXTERNAL PROCEDURE CADRE ;
EXTERNAL PROCEDURE CTEOL;
EXTERNAL PROCEDURE CTEOS;
EXTERNAL PROCEDURE HOME;
EXTERNAL PROCEDURE BELL         (N:INTEGER);
EXTERNAL PROCEDURE DESABLE;
EXTERNAL PROCEDURE REVERSE;
EXTERNAL PROCEDURE LOWINT;
EXTERNAL PROCEDURE POS  (L,C:INTEGER);


EXTERNAL PROCEDURE @HLT;


(MODULE GESMAP)
EXTERNAL FUNCTION  get_fl : CHAR;
EXTERNAL PROCEDURE load_map (filename:t_filename; var map:t_map);
EXTERNAL PROCEDURE ini_map  (map: t_map;code    :INTEGER);
EXTERNAL PROCEDURE read_map (map: t_map;var zone:INTEGER);
EXTERNAL PROCEDURE write_map(map: t_map;zone    :INTEGER);
```

```
FUNCTION valid_descr (n_anal,n_descr : INTEGER): BOOLEAN;

VAR     i       : INTEGER;

BEGIN
        IF n_descr = 9999
        THEN BEGIN
                valid_descr := true;
                EXIT;
             END;
        SEEKREAD(fildsc,n_descr);
        IF (IORESULT<>0) or (fildsc^.nro_d <> n_descr )
        THEN BEGIN
                erreur (6);
                valid_descr := FALSE;
                EXIT;
             END;
        SEEKREAD(filcom,n_descr);
        IF filcom^.list_analyse[1] < 27
        THEN BEGIN
                filcom^.list_analyse[1]  :=filcom^.list_analyse[1] + 1;
                filcom^.list_analyse[filcom^.list_analyse[1] ] := n_ana
                valid_descr := TRUE;
                SEEKWRITE(filcom,n_descr);
                IF IORESULT  <> 0
                THEN BEGIN
                        erreur(5);
                        EXIT;
                     END;
             END
        ELSE BEGIN
                erreur(0);
                valid_descr := FALSE;
              END;
END;
```

```
PROCEDURE temporisation;

VAR     c :     CHAR;

BEGIN
        pos(23,40);
        CASE screen_mode OF
        0,1 :reverse;
        2 :;
        ELSE;
        END;
        bell(2);
        WRITE('Tapez sur une touche pour continuer ');
        desable;
        READ(c);
END;



PROCEDURE cv_entier_en_date (d:INTEGER; VAR j,m,a:INTEGER);

BEGIN
        a := (d DIV 372) + 80;
        d := d MOD 372        ;
        m := (d DIV 31)  +1   ;
        j := d MOD 31         ;
END;



PROCEDURE cv_date_en_entier (VAR d:INTEGER; j,m,a:INTEGER);

BEGIN
        d := 372*(a-80) + (m-1)*31 + j ;
END;
```

```
PROCEDURE next_page(VAR nro : INTEGER;j,m,a:INTEGER) ;
BEGIN
        WRITELN(lpt,'                                                ',j:2,'/',
                m:2,'/',a:2,'                                    PAGE ',NRO:3);
        nro:=nro+1;
END;


PROCEDURE entete;

VAR i : INTEGER;

BEGIN
        WRITELN(lpt,'');
        WRITE  (lpt,'                    ');
        WRITELN(lpt,' LABORATOIRE MEDICAL ET ANALYSES BIOLOGIQUES.');
        WRITE  (lpt,'                    ');
        WRITELN(lpt,'   101, Chaussee de Binche.       7000 MONS');
        WRITELN(lpt,chr(14));
        FOR i :=1 TO 10 DO WRITELN(lpt);
END;


PROCEDURE stopper;
BEGIN
        home; cteos;
        pos(12,0);
        reverse;
        CLOSE(filcom,result);
        CLOSE(fildsc,result);
        CLOSE(filana,result);
        WRITELN ('FICHIERS SAUVES ...');
        WRITELN;
        IF oui ('FIN DE PROGRAMME : CONFIRMATION','r') THEN @HLT;
END;
```

```
PROCEDURE erreur(n:INTEGER);

VAR     i,j     : INTEGER;

BEGIN
        bell(5);
        pos(23,20);
        CASE n OF
    0 : WRITE(
        'ERREUR: LA LISTE D''ANALYSES DE CETTE DESCRIPTION EST PLEINE')
    1 : WRITE('ERREUR: IMPOSSIBILITE D''ACCEDER A CE FICHIER ');
    2 : WRITE('ERREUR: ENTREZ UN NOMBRE ENTRE 0 ET 3  !!!');
    3 : WRITE('ERREUR: ENTREZ UN NOMBRE ENTRE 0 ET 9  !!!');
    4 : WRITE('ERREUR: CETTE ANALYSE EXISTE DEJA !!!');
    5 : WRITE('ERREUR: L''OPERATION N''A PU ETRE ENREGISTREE !!');
    6 : WRITE('ERREUR: CETTE DESCRIPTION N''EXISTE PAS !!!');
    7 : WRITE('ERREUR: CETTE ANALYSE  N''EXISTE PAS !!!');
    8 : WRITE('ERREUR: CETTE DESCRIPTION EXISTE DEJA !!!');
    9 : WRITE('ERREUR: VALEUR INCONNUE !!!');
   10 : WRITE('PAS D''INFORMATION SUPPLEMENTAIRE DISPONIBLE !!')
        ELSE;
        END;
        FOR i:=1 TO MAXINT DO ;
        FOR i:=1 TO MAXINT DO ;
        pos(23,19);
        cteol;
END;


PROCEDURE MENU;
BEGIN
        home;
        cteos;
        menu_z_cour:=0;
        cadre;
        ini_map(menu_map,0);
        last_op := last_menu;
        read_map(menu_map,menu_z_cour);
        IF menu_z_cour < 0      THEN stopper;
        IF menu_z_cour >=  1000 THEN erreur (10);
        last_menu := last_op;
END;
```

```
PROCEDURE anal_ajouter;

LABEL 10;

VAR     nro_anal,bid     : INTEGER;
        ok               : BOOLEAN;

BEGIN
        ok := FALSE;
        anal_z_cour := 0;
        nro_anal := gl_integer;
        SEEKREAD(filana,nro_anal);
        IF (IORESULT=0) AND (filana^.nro_a = nro_anal)
        THEN BEGIN
                erreur(4);
                EXIT;
             END;
        home; cteos;
        ini_map (anal_map,screen_mode);
        write_map(anal_map,1);
        filana^.nro_a := nro_anal;
        REPEAT
     10:read_map(anal_map,anal_z_cour);
        IF anal_z_cour > 1000 THEN BEGIN
                                      anal_z_cour := anal_z_cour -1000
                                      help(anal_z_cour+100);
                                      GOTO 10;
                                   END;
        CASE anal_z_cour OF
        2: filana^.libelle := gl_str25;
        3: BEGIN
                filana^.nro_descr:= gl_integer;
                IF NOT valid_descr (nro_anal,filana^.nro_descr)
                        THEN GOTO 10;
            END;
        -1: ok := TRUE
        ELSE;
        END;
        IF NOT ok THEN anal_z_cour := -anal_z_cour ;
        UNTIL ok = TRUE;
        SEEKWRITE(filana,nro_anal);
        IF IORESULT  <> 0
        THEN BEGIN
                erreur(5);
                EXIT;
             END;

END;
```

```
PROCEDURE anal_detruire;

VAR     nro_anal        : INTEGER;

BEGIN
        nro_anal := gl_integer;
        SEEKREAD (filana,nro_anal);
        IF (IORESULT<>0) OR (filana^.nro_a <> nro_anal)
        THEN BEGIN erreur(7); EXIT; END;
        filana^.nro_a := -1;
        SEEKWRITE(filana,nro_anal);
        IF IORESULT <> 0
        THEN BEGIN erreur(5); EXIT; END;
END;


PROCEDURE anal_consulter;

VAR     nro_anal        : INTEGER;

BEGIN
        nro_anal := gl_integer;
        SEEKREAD (filana,nro_anal);
        IF (IORESULT<>0) OR (filana^.nro_a <> nro_anal)
        THEN BEGIN erreur(7); EXIT; END;
        home; cteos;
        ini_map(anal_map,screen_mode);
        gl_integer := nro_anal ;
        write_map(anal_map,1);
        gl_str25 := filana^.libelle;
        write_map(anal_map,2);
        gl_integer := filana^.nro_descr;
        write_map(anal_map,3);
        pos(10,1);
        IF oui (
         'VOULEZ-VOUS LA DESCRIPTION COMPLETE DE CETTE ANALYSE','N')
        THEN BEGIN
                pos(10,1);      cteos;
                desc_consulter;
                temporisation;
             END
        ELSE BEGIN pos(10,1); cteol; END;

END;
```

```
PROCEDURE anal_modifier;

LABEL 20;

VAR     nro_anal          : INTEGER;
        modif,ok          : BOOLEAN;

BEGIN
        ok:=FALSE;
        nro_anal := gl_integer;
        SEEKREAD (filana,nro_anal);
        IF (IORESULT<>0) OR (filana^.nro_a <> nro_anal)
        THEN BEGIN erreur(7); EXIT; END;
        filana^.nro_a := nro_anal;
        home; cteos;
        ini_map(anal_map,screen_mode);
        anal_z_cour := 0;
        gl_integer := nro_anal ;
        write_map(anal_map,1);
        gl_str25 := filana^.libelle;
        write_map(anal_map,2);
        gl_integer := filana^.nro_descr;
        write_map(anal_map,3);
        REPEAT
     20:read_map(anal_map,anal_z_cour);
        IF anal_z_cour > 1000 THEN BEGIN
                                        anal_z_cour := anal_z_cour -100
                                        help(anal_z_cour+100);
                                        GOTO 20;
                                END;
        CASE anal_z_cour OF
        2: filana^.libelle := gl_str25;
        3: BEGIN
                filana^.nro_descr := gl_integer;
                IF NOT valid_descr (nro_anal,filana^.nro_descr)
                        THEN GOTO 20;
            END;
        -1: ok := TRUE
        ELSE;
        END;
        IF NOT ok THEN anal_z_cour := -anal_z_cour ;
        UNTIL ok = TRUE;
        SEEKWRITE(filana,nro_anal);
        IF IORESULT <> 0
        THEN BEGIN erreur(5); EXIT; END;
END;
```

```
PROCEDURE desc_ajouter;

LABEL 1;

VAR     n_desc,n          : INTEGER;
        ok                : BOOLEAN;
        c                 : CHAR   ;
        i                 : INTEGER;
        res               : INTEGER;

PROCEDURE ini;
BEGIN
WITH fildsc^ DO
BEGIN
        home; cteos;
        prelevement := 'D';
        dispatching := 'I';
        technique   := 'D';
        qte_utilisee:= 0;
        dbl_rincage := FALSE;
        nro1_tampon := MAXINT ;
        nro2_tampon := MAXINT ;
        qte1_tampon := MAXINT ;
        qte2_tampon := MAXINT ;
        filcom^.txt := '                          ';
        filcom^.list_analyse[1]:=1;
END;
END;

BEGIN
WITH fildsc^ DO
BEGIN
        n_desc := gl_integer;
        SEEKREAD(fildsc,n_desc);
        res := IORESULT;
        IF menu_z_cour = menu_a_d
        THEN BEGIN
                IF (res=0)AND(fildsc^.nro_d = n_desc )
                THEN BEGIN
                        erreur(8);
                        EXIT;
                     END
             END
        ELSE
                IF (res <> 0) OR (fildsc^.nro_d <> n_desc )
                THEN    EXIT;
```

```
IF menu_z_cour=menu_a_d
 THEN BEGIN
         ini;
         inimap(desc_map,screen_mode);
         write_map(desc_map,1);
      END;
 ok := FALSE;
 filcom^.date:= date_du_jour;
 cv_entier_en_date(filcom^.date,jour,mois,annee);
 write_map(desc_map,14);
 nro_d := n_desc;
 desc_z_cour := 0;
 REPEAT
 read_map (desc_map,desc_z_cour);
1:IF desc_z_cour>=1000 THEN BEGIN desc_z_cour := desc_z_cour -100
                                  help(desc_z_cour);
                                  read_map (desc_map,desc_z_cour)
                                  goto 1;
                           END;
 CASE desc_z_cour OF
 2: BEGIN
         IF gl_car IN set_prelevement
          THEN prelevement := gl_car
          ELSE BEGIN  erreur(9);
                      read_map(desc_map,desc_z_cour);
                      GOTO 1;
               END;
         wr_prelevement(ecran);
      END;
 3: qte_utilisee := gl_integer;
 5: BEGIN
         IF gl_car IN set_dispatching
          THEN dispatching := gl_car
          ELSE BEGIN  erreur(9);
                      read_map(desc_map,desc_z_cour);
                      GOTO 1;
               END;
         wr_dispatching (ecran);
      END;
 6: BEGIN
         IF gl_car IN set_technique
          THEN technique := gl_car
          ELSE BEGIN  erreur(9);
                      read_map(desc_map,desc_z_cour);
                      GOTO 1;
               END;
          wr_technique (ecran);
      END;
```

```
        7: BEGIN
                IF gl_car IN ['o','n','O','N']
                 THEN BEGIN IF (gl_car='o') OR (gl_car='O')
                                THEN dbl_rincage := TRUE
                                ELSE dbl_rincage := FALSE;
                      END
                 ELSE BEGIN  erreur(9);
                             read_map(desc_map,desc_z_cour);
                             GOTO 1;
                      END;
                wr_dbl_rincage (ecran);
             END;
         8:  nro1_tampon := gl_integer;
         9:  qte1_tampon := gl_integer;
         11: nro2_tampon := gl_integer;
         12: qte2_tampon := gl_integer;
         15: filcom^.txt := gl_str70;
         -1 : ok := TRUE
         ELSE;
         END;
         IF NOT ok THEN desc_z_cour := -desc_z_cour ;
         UNTIL ok;
         SEEKWRITE(filcom,n_desc);
         IF IORESULT <> 0
         THEN BEGIN
                erreur(5);
                EXIT;
             END;
         SEEKWRITE(fildsc,n_desc);
         IF IORESULT <> 0
         THEN BEGIN
                erreur(5);
                EXIT;
             END;
END;
END;
```

```
PROCEDURE wr_prelevement (ecran:BOOLEAN);
BEGIN
        CASE fildsc^.prelevement OF
        's','S':gl_str25 := 'SANG                ';
        'u','U':gl_str25 := 'URINE               ';
        'd','D':gl_str25 := 'DIVERS              ';
        'f','F':gl_str25 := 'SELLES              ';
        'g','G':gl_str25 := 'LIQUIDE GASTRIQUE   ';
        'p','P':gl_str25 := 'PONCTION            ';
        'a','A':gl_str25 := 'LIQUIDE AMNIOTIQUE  ';
        'l','L':gl_str25 := 'L.C.R               '
        ELSE;
        END;
        IF ecran THEN write_map(desc_map,2) ELSE WRITE(lpt,gl_str25);
END;

PROCEDURE wr_dispatching (ecran:BOOLEAN);
BEGIN
        CASE fildsc^.dispatching OF
        'i','I': gl_str25 := 'IMMEDIAT           ';
        'b','B': gl_str25 := 'BATCH              '
        ELSE; END;
        IF ecran THEN write_map(desc_map,5) ELSE WRITE(lpt,gl_str25);
END;

PROCEDURE wr_technique (ecran:BOOLEAN);
BEGIN
        CASE fildsc^.technique OF
        'I','i': gl_str25 := 'ISOTOPES             ';
        'D','d': gl_str25 := 'DIVERS               '
        ELSE; END;
        IF ecran THEN write_map(desc_map,6) ELSE WRITE(lpt,gl_str25);
END;

PROCEDURE wr_dbl_rincage (ecran:BOOLEAN);
BEGIN
        IF fildsc^.dbl_rincage
          THEN gl_str25 := 'OUI' ELSE gl_str25 := 'NON';
        IF ecran THEN write_map(desc_map,7) ELSE WRITE(lpt,gl_str25);
END;

PROCEDURE wr_list_analyse;
VAR i :INTEGER;
BEGIN
        FOR i :=1 TO 27 DO gl_tab27[i] := filcom^.list_analyse[i];
        write_map(desc_map,16);
END;
```

```
PROCEDURE wr_integer(n:INTEGER);
BEGIN
        IF n< MAXINT THEN  WRITE(lpt,n:4)
                     ELSE WRITE(lpt,'    ');
END;


PROCEDURE help ( n : INTEGER);
BEGIN
        pos(20,1);
        WRITE('----------------------------------------');
        WRITE('----------------------------------------');
        pos(21,1);
        CASE n OF
   2: BEGIN
      WRITE('LIQ. AMNIOTIQUE = A, SANG     = S, URINE  = U, DIVERS = D'
      pos(22,1);
      WRITE('LIQ. GASTRIQUE  = G, PONCTION = P, SELLES = F, L.C.R. = L'
      END;

   3: WRITE(
        'QUANTITE DE PRELEVEMENT UTILISEE  PAR ANALYSE (en micro-L.)');
   5: WRITE(
        'DISPATCHING ... IMMEDIAT = I,  BATCH = B');
   6: WRITE(
        'TECHNIQUE UTILISEE ... ISOTOPES = I , DIVERS = D');
   7: WRITE('DOUBLE RINCAGE :OUI = O   , NON =N');

   8: WRITE('NUMERO DU TAMPON 1  UTILISE ');

   9: WRITE('QUANTITE DE TAMPON 1 UTILISEE PAR ANALYSE (en micro-L.)');

   11: WRITE('NUMERO DU TAMPON 2  UTILISE ');

   12: WRITE('QUANTITE DE TAMPON 2 UTILISEE PAR ANALYSE (en micro-L.)')

   15:WRITE('COMMENTAIRE (max 70 carac)');

   16:WRITE(
     'NUMERO DES ANALYSES POUVANT ETRE DECRITES PAR CETTE DESCRIPTION');
  102:WRITE('LIBELLE DE L''ANALYSE (en 25 car maximum) ');

  103:WRITE('NUMERO DE LA DESCRIPTION CORRESPONDANT A CETTE ANALYSE ')

        ELSE; END;
        temporisation;
        pos(20,0); cteos;
END;
```

```
PROCEDURE desc_detruire;

VAR     n_desc  : INTEGER;

BEGIN
        n_desc := gl_integer;
        SEEKREAD (fildsc,n_desc);
        IF (IORESULT<>0) OR (fildsc^.nro_d <> n_desc)
        THEN BEGIN
                erreur(6);
                EXIT;
             END;
        fildsc^.nro_d := -1;
        SEEKWRITE(fildsc,n_desc);
        IF IORESULT <> 0
        THEN BEGIN
                erreur(5);
                EXIT;
             END;
END;


PROCEDURE desc_modifier;

VAR     n_desc          : INTEGER;

BEGIN
        desc_consulter;
        desc_ajouter;
END;
```

```
PROCEDURE desc_consulter;

VAR     n_desc,i          : INTEGER;

BEGIN
with fildsc^ DO
BEGIN
        IF (menu_z_cour = menu_c_d) OR (menu_z_cour = menu_m_d)
                THEN n_desc := gl_integer
                ELSE n_desc := filana^.nro_descr;
        SEEKREAD (filcom,n_desc);
        SEEKREAD (fildsc,n_desc);
        IF (IORESULT<>0) OR (fildsc^.nro_d <> n_desc)
        THEN BEGIN
                erreur(6);
                EXIT;
             END;
        IF (menu_z_cour = menu_c_d) OR (menu_z_cour = menu_m_d)
        THEN BEGIN home; cteos; END;
        inimap(desc_map,screen_mode);
        gl_integer := n_desc;
        write_map(desc_map,1);
        wr_prelevement (ecran);
        gl_integer := qte_utilisee;
        write_map(desc_map,3);
        wr_dispatching (ecran);
        wr_technique (ecran);
        wr_dbl_rincage (ecran);
        gl_integer := nro1_tampon;
        write_map(desc_map,8);
        gl_integer := qte1_tampon;
        write_map(desc_map,9);
        gl_integer := nro2_tampon;
        write_map(desc_map,11);
        gl_integer := qte2_tampon;
        write_map(desc_map,12);
        cv_entier_en_date(filcom^.date,jour,mois,annee);
        write_map(desc_map,14);
        gl_str70 := filcom^.txt;
        write_map(desc_map,15);
        wr_list_analyse;
        gl_integer := n_desc;
        IF menu_z_cour = menu_c_d THEN temporisation;
END;
END;
```

```
PROCEDURE liste1;

VAR     i,cpt           : INTEGER;
        nro_page        : INTEGER;
        j,m,a           : INTEGER;
        n_enre          : INTEGER;
        com             : t_commentaire                         ;
        ana             : t_analyse                             ;
        dsc             : t_description                         ;

PROCEDURE imp1;
BEGIN
WITH dsc DO
BEGIN
        cv_entier_en_date(com.date,jour,mois,annee);
        IF (jour<=0) OR (mois<=0) OR (annee<=0)
        OR (NOT (dispatching IN set_dispatching))
        OR (NOT (prelevement IN set_prelevement))
        OR (NOT (technique   IN set_technique  ))
        OR (nro_d<0) OR (nro_d>1000)
        OR (n_enre <> nro_d)    THEN EXIT;
        IF cpt <9 THEN BEGIN cpt:= 59; next_page(nro_page,j,m,a); END;
        pos (5,35); WRITE('DESCR #',nro_d:4);
        WRITELN(lpt);
        WRITELN(lpt,'DESC#',nro_d:4,' ',com.txt);
        WRITELN(lpt);
        WRITE(lpt,'DATE: ',jour:2,'/',mois:2,'/',annee:2,
                  '      DISPATCHING: ');
        wr_dispatching (fichier);
        WRITELN(lpt);   WRITELN(lpt);
        WRITE(lpt,'ANALYSES: ');
        I := 2;
        WHILE (i <= com.list_analyse[1] ) AND (i<=27) DO
        BEGIN
        WRITE(lpt,com.list_analyse[i]:4);
        IF i= 16 THEN BEGIN
                        WRITELN(lpt);
                        WRITELN(lpt);
                        cpt:=cpt-2;
                      END;
        i := i +1;
        END;
        WRITELN(lpt);
        WRITE  (lpt,'_____________________________________');
        WRITELN(lpt,'_____________________________________');
        cpt:=cpt-7;
END;
END;
```

```
BEGIN
        ASSIGN(lpt,'b:liste1.lpt');
        REWRITE(lpt);
        home;
        cteos;
        pos(10,10);
        WRITE ('CREATION DU FICHIER "B:LISTE1.LPT"  EN COURS ....');
        cv_entier_en_date(date_du_jour,j,m,a);
        cpt:=0; n_enre := 0;
        nro_page := 1;
        entete;
        WRITELN(lpt,'                    LISTE');
        WRITELN(lpt);
        WRITELN(lpt,'                     DES');
        WRITELN(lpt);
        WRITELN(lpt,'                DESCRIPTIONS');
        WRITELN(lpt);
        WRITELN(lpt,'                 D''ANALYSE');
        WRITELN(lpt,chr(15));
        CLOSE (fildsc,result);
        CLOSE (filcom,result);
        RESET (fildsc);
        RESET (filcom);
        WHILE NOT EOF(fildsc) DO
        BEGIN
                READ (fildsc,dsc);
                READ (filcom,com);
                imp1;
                n_enre := n_enre +1;
        END;
        pos(13,10);
        WRITE ('CREATION : [OK]');
        CLOSE (lpt,result);
        CLOSE (fildsc,result);
        CLOSE (filcom,result);
        RESET (fildsc);
        RESET (filcom);
        temporisation;

END;
```

```
PROCEDURE liste2;

VAR     i,cpt           : INTEGER;
        nro_page        : INTEGER;
        j,m,a           : INTEGER;
        n_enre          : INTEGER;
        com             : t_commentaire                         ;
        ana             : t_analyse                             ;
        dsc             : t_description                         ;

PROCEDURE imp2;
BEGIN
WITH dsc DO
BEGIN
        cv_entier_en_date(com.date,jour,mois,annee);
        IF (jour<=0) OR (mois<=0) OR (annee<=0)
        OR (NOT (dispatching IN set_dispatching))
        OR (NOT (prelevement IN set_prelevement))
        OR (NOT (technique   IN set_technique  ))
        OR (nro_d<0) OR (nro_d>1000)
        OR (n_enre <> nro_d)    THEN EXIT;
        pos (5,35); WRITE('DESCR #',nro_d:4);
        cpt:= 59; next_page(nro_page,j,m,a);
        WRITELN(lpt);   WRITELN(lpt);
        WRITELN(lpt,chr(14));
        WRITELN(lpt,'DESCRIPTION #',nro_d:4);
        WRITELN(lpt,chr(15));
        WRITELN(lpt);
        WRITELN(lpt);
        WRITELN(lpt,com.txt);
        WRITELN(lpt);
        WRITE  (lpt,'PRELEVEMENT   : ');
        wr_prelevement (fichier);
        WRITELN(lpt);
        WRITELN(lpt);
        WRITELN(lpt,'QTE UTILISEE  : ',qte_utilisee:4,' micro-L.');
        WRITELN(lpt);
        WRITE  (lpt,'DISPATCHING   : ');
        wr_dispatching (fichier);
        WRITELN(lpt);
        WRITELN(lpt);
        WRITE  (lpt,'DBLE RINCAGE  : ');
        wr_dbl_rincage (fichier);
        WRITELN (lpt);
        WRITELN(lpt);
        WRITE  (lpt,'TECHNIQUE     : ');
        wr_technique (fichier);
```

```
        WRITELN (lpt);
         WRITELN(lpt);
         WRITE  (lpt,'NRO TAMPON 1   : ');wr_integer(nro1_tampon);
         WRITE  (lpt,'                      QUANTITE : ');
         wr_integer(qte1_tampon); WRITELN(lpt,' micro-L.');
         WRITELN(lpt);
         WRITE  (lpt,'NRO TAMPON 2   : ');wr_integer(nro2_tampon);
         WRITE  (lpt,'                      QUANTITE : ');
         wr_integer(qte2_tampon); WRITELN(lpt,' micro-L.');
         WRITELN(lpt);
         WRITELN(lpt,'DATE           : ',jour:2,'/',mois:2,'/',annee:2);

         WRITELN(lpt);
         WRITELN(lpt,'ANALYSES       :');

         i := 2;
         WHILE (i <= com.list_analyse[1] ) AND (i<=27) DO
         BEGIN
         WRITE  (lpt,'                      ');
         WRITE  (lpt,com.list_analyse[i]:4);
         SEEKREAD(filana,com.list_analyse[i]);
         IF (IORESULT<>0) OR (com.list_analyse[i]<>ana.nro_a)
         THEN WRITELN(lpt,'  inconnue !!')
         ELSE WRITELN(lpt,'  ',ana.libelle);
         i := i +1;
         END;
         WRITELN(lpt);
END;
END;
```

```
BEGIN
        ASSIGN(lpt,'b:liste2.lpt');
        REWRITE(lpt);
        home;
        cteos;
        pos(10,10);
        WRITE ('CREATION DU FICHIER "B:LISTE2.LPT"  EN COURS ....');
        cv_entier_en_date(date_du_jour,j,m,a);
        cpt:=0;  n_enre := 0;
        nro_page := 1;
        entete;
        WRITELN(lpt,'                    DETAIL');
        WRITELN(lpt);
        WRITELN(lpt,'                     DES');
        WRITELN(lpt);
        WRITELN(lpt,'                  DESCRIPTIONS');
        WRITELN(lpt);
        WRITELN(lpt,'                   D''ANALYSE');
        WRITELN(lpt,chr(15));
        CLOSE (fildsc,result);
        CLOSE (filcom,result);
        RESET (fildsc);
        RESET (filcom);
        WHILE NOT EOF(fildsc) DO
        BEGIN
                READ (fildsc,dsc);
                READ (filcom,com);
                imp2;
                n_enre := n_enre +1;
        END;
        pos(13,10);
        WRITE ('CREATION : [OK]');
        CLOSE (fildsc,result);
        CLOSE (filcom,result);
        RESET (fildsc);
        RESET (filcom);
        CLOSE (lpt,result);
        temporisation;

END;
```

```
PROCEDURE liste3;

VAR     i,cpt               : INTEGER;
        nro_page            : INTEGER;
        j,m,a               : INTEGER;
        n_enre              : INTEGER;
        com                 : t_commentaire                    ;
        ana                 : t_analyse                        ;
        dsc                 : t_description                    ;

PROCEDURE imp3;

VAR     k : INTEGER;

BEGIN
WITH dsc DO
BEGIN
        cv_entier_en_date(com.date,jour,mois,annee);
        IF (jour<=0) OR (mois<=0) OR (annee<=0)
        OR (NOT (dispatching IN set_dispatching))
        OR (NOT (prelevement IN set_prelevement))
        OR (NOT (technique   IN set_technique  ))
        OR (ana.nro_a<0) OR (ana.nro_a>1000)
        OR (n_enre <> ana.nro_a)
                        THEN EXIT;
        pos (5,35); WRITE('ANALYSE #',ana.nro_a:4);
        IF cpt <11 THEN BEGIN cpt:= 59; next_page(nro_page,j,m,a); END;
        SEEKREAD(filcom,ana.nro_descr);
        SEEKREAD(fildsc,ana.nro_descr);
```

```
        IF (IORESULT <> 0) OR (ana.nro_descr <> nro_d)
         THEN BEGIN
                 WRITELN(lpt);
                 WRITELN(lpt,'ANALYSE :',ana.nro_a:4,
                 '  pas de description #',ana.nro_descr);
                 WRITELN(lpt);              cpt := cpt -4;
                 WRITE  (lpt,'_____________________________________');
                 WRITELN(lpt,'_____________________________________');
                 EXIT;
              END;
         WRITELN(lpt);
         WRITE(lpt,'ANALYSE     #',ana.nro_a:4,'     ',ana.libelle);
         FOR k := 1 TO (25-length(ana.libelle)) DO WRITE(lpt,' ');
         WRITELN(lpt,'           DATE : ',jour:2,'/',mois:2,'/',annee:2)
         WRITELN(lpt);
         WRITE  (lpt,'DESCRIPTION #',ana.nro_descr:4);
         WRITE  (lpt,'    DISPATCHING : ');
         wr_dispatching (fichier);
         WRITE  (lpt,'  TECHNIQUE : ');
         wr_technique (fichier);
         WRITELN (lpt);  WRITELN (lpt);
         WRITE  (lpt,'NRO TAMPON 1  : ');wr_integer(nro1_tampon);
         WRITE  (lpt,'                    QUANTITE : ');
         wr_integer(qte1_tampon); WRITELN(lpt,' micro-L.');

         WRITELN(lpt);
         WRITE  (lpt,'NRO TAMPON 2  : ');wr_integer(nro2_tampon);
         WRITE  (lpt,'                    QUANTITE : ');
         wr_integer(qte2_tampon); WRITELN(lpt,' micro-L.');
         WRITELN(lpt);   cpt:=cpt-11;
         WRITE  (lpt,'_____________________________________');
         WRITELN(lpt,'_____________________________________');
END;
END;
```

```
BEGIN
        ASSIGN(lpt,'b:liste3.lpt');
        REWRITE(lpt);
        home;
        cteos;
        pos(10,10);
        WRITE ('CREATION DU FICHIER "B:LISTE3.LPT"  EN COURS ....');

        cv_entier_en_date(date_du_jour,j,m,a);
        cpt:=0;  n_enre := 0;
        nro_page := 1;
        entete;
        WRITELN(lpt,'                    LISTE');
        WRITELN(lpt);
        WRITELN(lpt,'                     DES');
        WRITELN(lpt);
        WRITELN(lpt,'                   ANALYSES');
        WRITELN(lpt,chr(15));
        CLOSE (filana,result);
        RESET (filana);
        WHILE NOT EOF(filana) DO
        BEGIN
                READ (filana,ana);
                imp3;
                n_enre := n_enre +1;
        END;

        pos(13,10);
        WRITE ('CREATION : [OK]');
        CLOSE (lpt,result);
        CLOSE (filana,result);
        RESET (filana);
        temporisation;

END;
```

```
{  PROGRAMME PRINCIPAL }

BEGIN
        (>>> INITIALISATIONS <<<)
        last_menu := -1;
        ecran := TRUE;  fichier := FALSE;
        home;   cteos;  pos(12,30);
        WRITE ('Donnez la date s.v.p ');
        pos(14,30);
        jour := inputint('jour  (ex:27) :','R');
        pos(15,30);
        mois := inputint('mois  (ex:07) :','R');
        pos(16,30);
        annee := inputint('annee (ex:83) :','R');
        cv_date_en_entier(date_du_jour,jour,mois,annee);
        set_technique    := ['i','I','d','D']    ;
        set_prelevement := ['s','u','d','f','g','p','a','l',
                            'S','U','D','F','G','P','A','L']    ;
        set_dispatching := ['i','I','b','B']    ;
        load_map ('b:menu.map'  ,menu_map);
        load_map ('b:descript.map',desc_map);
        load_map ('b:analyse.map',anal_map);
        REPEAT
        pos(20,1);cteol;bell(2);
        screen_mode := inputint
        ('MODE D''ECRAN: 0=normal; 1=reverse; 2=faible intensite :','R'
        UNTIL (screen_mode=0) OR (screen_mode=1) OR (screen_mode=2) ;
        home;   cteos;  pos(12,0);
        IF oui ('CREATION DE 3 NOUVEAUX FICHIERS ','R')
        THEN BEGIN
                WRITELN;
                IF oui
                 ('CONFIRMEZ LA DESTRUCTION DES ANCIENS FICHIERS ','R')
                        THEN BEGIN
                        WRITELN;
                        ASSIGN (filcom,'b:filcom.dat');
                        ASSIGN (fildsc,'b:fildsc.dat');
                        ASSIGN (filana,'b:filana.dat');
                        purge(filcom);
                        purge(fildsc);
                        purge(filana);
                        REWRITE(filcom);
                        REWRITE(fildsc);
                        REWRITE(filana);
                        CLOSE  (filcom,result);
                        CLOSE  (fildsc,result);
                        CLOSE  (filana,result);
                        END;
           END;
```

```
        ASSIGN (filcom,'b:filcom.dat');
         ASSIGN (fildsc,'b:fildsc.dat');
         ASSIGN (filana,'b:filana.dat');
         RESET(filcom);
         RESET(fildsc);
         RESET(filana);

         REPEAT
         MENU;
         CASE menu_z_cour OF
         menu_a_a : anal_ajouter;
         menu_d_a : anal_detruire;
         menu_c_a : anal_consulter;
         menu_m_a : anal_modifier;
         menu_a_d : desc_ajouter;
         menu_d_d : desc_detruire;
         menu_c_d: desc_consulter;
         menu_m_d: desc_modifier;
         menu_lst1: liste1;
         menu_lst2: liste2;
         menu_lst3: liste3;
         menu_stop: stopper;
         END;
         UNTIL FALSE ;
END.
```

Chapter 12

Programme de dispatching batch.

12.1 Routines de gestion des groupes d'analyses.

12.1.1 Signification des principales variables.

descripion d'un groupe d'analyses:

```
t_groupe_ana     = RECORD
        nbr_ana          : INTEGER ;     nombre d'analyses dans ce groupe
        tab_ana          : t_tab_int ;   liste des numéros d'analyse
        tab_p_qte        : t_tab_int ;   liste des quantités de prélèvemnt
        tab_t_nro        : t_tab_int ;   liste des numéros de tampon
        tab_t_qte        : t_tab_int ;   liste des quantités de tampon
        tab_dbl_rcg      : t_tab_bool ;  liste des codes de double rincage
        position         : INTEGER ;     première colonne à utiliser
        nb_col           : INTEGER ;     nombre de colonnes a utiliser
        option           : CHAR ;        code option
  END;

t_tab_groupe     = ARRAY [1..max_groupe] OF t_groupe_ana;
                   table de groupes d'analyse.
```

12.1.2 Description des fonctions.

cadre1 desine un cadre à l'écran

PROCEDURE get_groupe (VAR tab_groupe: t_tab_groupe; VAR nb_groupe: INTEGER);
Cette procédure renvoie dans "tab_grp", les données relatives au "nb_groupes" introduits par l'utilisateur et dans "nb_groupes" le nombre de groupes introduits.

PROCEDURE lire_groupe (var groupe: t_groupe_ana; nro: integer; var stop: boolean);
effectue la gestion d'écran nécessaire à l'introduction d'un groupe.

PROCEDURE error (nro_err, n1, n2: INTEGER);
affiche le message d'erreur correspondant à nro_err en utilisant éventuellement les valeurs de n1 et/ou n2

PROCEDURE valid_grp (VAR groupe: t_groupe_ana; nro: INTEGER);
Cette procédure permet de vérifier certaines incohérences dans les données introduites par l'utilisateur.Si on se souvient que les quantités necéssaires pour des analyses introduites dans un même groupe, sont devidées au même endroit, il va de soi que les conditions suivantes doivent être respectées au niveau des analyses d'un même groupe:

1. Le type de dispatching de toutes les analyses doit être "batch".

2. Si l'option choisie est "tampon" ou "dilution", tous les tampons utilisés pour ces analyses doivent être identiques, et de plus, le rapport qte tampon / qte prélèvement doit être constant.

3. Toutes les analyses ainsi que leur description doivent être connues.

Si une des erreurs mentionnées ci-dessus survient, le demande de dispatching de ce groupe est annulée. Cela est signalé à l'utilisateur par un message. Au niveau programme, le nombre d'analyses de ce groupe est mis à 0 ce qui évite tout traitement ultérieur. La procédure assure une deuxième fonction: une fois le groupe validé, elle se charge pour chaque analyse du groupe de remplir les zones suivantes:

- tab_p_qte (tableau des quantités de prélèvement)

- tab_t_nro (tableau des numéros de tampon)

- tab_t_qte (tableau des quantités de tampon)

- tab_dbl-rcg (tableau des booléens de double rincage)

Pour chaque analyse tab_ana[i],(1<=i<=nbr_ana) la procédure remplit tab_p_qte[i],tab_t_nro[i],tab_t_qte[i],tab_dbl_rcg[i]

à partir de la description correspondant à l'analyse en question.

PROCEDURE valider (VAR tab_groupe: t_tab_groupe; VAR nb_groupe: INTEGER);
vérifie si toutes les contraintes sur tous les groupes introduits sont respectées

12.1.3 Texte source.

```
MODULE gestion_des_groupes;
CONST   max_ana_simul    =  40;
        max_groupe       =   5;
        max_ref          = 500;

TYPE    t_tab_int        = ARRAY [1..max_ana_simul] OF INTEGER;
        t_tab_bool       = ARRAY [1..max_ana_simul] OF BOOLEAN;

        t_groupe_ana     = RECORD
                               nbr_ana          : INTEGER ;
                               tab_ana          : t_tab_int ;
                               tab_p_qte        : t_tab_int ;
                               tab_t_nro        : t_tab_int ;
                               tab_t_qte        : t_tab_int ;
                               tab_dbl_rcg      : t_tab_bool ;
                               position         : INTEGER ;
                               nb_col           : INTEGER ;
                               option           : CHAR ;
                           END;

        t_tab_groupe     = ARRAY [1..max_groupe] OF t_groupe_ana;
        tab27    = ARRAY [1..27] OF INTEGER

        t_analyse =
                RECORD
                        nro_a            : INTEGER
                        libelle          : STRING[25]
                        nro_descr        : INTEGER
                END;

        t_description =
                RECORD
                        nro_d            : INTEGER
                        prelevement      : CHAR
                        technique        : CHAR
                        dispatching      : CHAR
                        dbl_rincage      : BOOLEAN
                        qte_utilisee     : INTEGER
                        nro1_tampon      : INTEGER
                        qte1_tampon      : INTEGER
                        nro2_tampon      : INTEGER
                        qte2_tampon      : INTEGER
                END;
```

```
        t_commentaire =
                RECORD
                        date            : INTEGER
                        txt             : STRING[70]
                        list_analyse    : tab27
                END;


VAR
        filcom          : EXTERNAL FILE OF t_commentaire
        filana          : EXTERNAL FILE OF t_analyse
        fildsc          : EXTERNAL FILE OF t_description


EXTERNAL PROCEDURE home;
EXTERNAL PROCEDURE cadre;
EXTERNAL PROCEDURE graphic;
EXTERNAL PROCEDURE desable;
EXTERNAL PROCEDURE pos(l,c:INTEGER);
EXTERNAL PROCEDURE bell(n:INTEGER);
EXTERNAL PROCEDURE cteos;
EXTERNAL FUNCTION oui (s:STRING;c:CHAR):BOOLEAN;


PROCEDURE cadre1;

VAR i,j         : INTEGER ;

BEGIN
home; cteos; cadre; graphic;
pos(2,0); WRITE  ('5000000000000000000000000000000000000000');
WRITELN('0000000000000000000000000000000000000000000:');
pos(19,0); WRITE  ('U                                                  ');
WRITELN('                                              j');
desable; home; pos(1,1);;
end;
```

```
PROCEDURE get_groupe (VAR tab_groupe:t_tab_groupe;VAR nb_groupe:INTEGER

VAR
        i,k,l,pmin,pmax : INTEGER ;
        stop            : BOOLEAN ;

PROCEDURE lire_groupe (var groupe:t_groupe_ana;nro:integer;
                       var stop:boolean);

VAR     ligne,col,n,c   : INTEGER ;

begin
        stop := false;
        cadre1;
        pos(1,27);
        write ('GROUPE D''ANALYSE #',NRO:2);
        repeat
        pos(10,11);
        write('OPTION ? (t,d,p) ou (s) pour stopper : ');
        bell(2);
        read (groupe.option);
        if groupe.option in ['s','S'] then begin stop := true; exit; en
        until groupe.option in ['t','T','d','D','p','P'] ;

        repeat
        pos(11,11);
        write('POSITION (entre ',pmin:2,' et ',pmax:2,') : ');
        bell(2);
        read (groupe.position);
        until (groupe.position <= pmax) and (groupe.position >= pmin);
        pmin := groupe.position;
```

```
        repeat
         pos(12,11);
         c:=pmax-pmin+1;
         write('NOMBRE DE COLONNES (entre 1 et ',c:2,') : ');
         bell(2);
         read (groupe.nb_col);
         until (groupe.nb_col <= c) and (groupe.position >= 1);
         pmin := pmin+groupe.nb_col;

         pos(21,3);
         WRITE('OPTION : ');
         case groupe.option of
         'd','D': write('DILUTION');
         't','T': write('TAMPON UNIQUEMENT');
         'p','P': write('PRELEVEMENT UNIQUEMENT')
         ELSE;
         END;
         pos(21,37);
         WRITE('POSITION : ',groupe.position:2);
         pos(21,56);
         WRITE('NBRE DE COLONNES : ',groupe.nb_col:2);

         pos(10,11);write('                                                    ');
         pos(11,11);write('                                                    ');
         pos(12,11);write('                                                    ');
         pos(5,2);  write('ANALYSES:');
         n:=0; ligne := 6; col := 11;
         repeat
                 n := n + 1;
                 groupe.tab_p_qte[n] := 0;
                 groupe.tab_p_qte[n] := 0;
                 if ligne = 16 then begin ligne := 6; col := col+ 10; e
                 pos(ligne,col);
                 readln (groupe.tab_ana[n]);
                 ligne := ligne + 1;
         until (groupe.tab_ana[n] =  -1) or (n = max_ana_simul);
         if (groupe.tab_ana[n] = -1)
         then begin pos(ligne-1,col); write ('       '); end;
         if (groupe.tab_ana[n] = -1)
         then groupe.nbr_ana := n-1
         else groupe.nbr_ana := n ;

end;
```

```
begin
        pmin:=1; pmax :=19;
        i := 0;
        stop := false;
        while (i<max_groupe) and (not stop) do
        begin
                i := i + 1;
                lire_groupe(tab_groupe[i],i,stop);
                pos (23,60);
                bell (1);
                if not stop then if  not oui('OK  ','n') then i := i-1;
        end;
        if stop then nb_groupe := i-1 else nb_groupe := i;

end;


PROCEDURE error (nro_err,n1,n2: INTEGER);
begin
   bell (2);
   CASE nro_err of
   0 :WRITE('ERREUR: ANALYSE #',n2:3,' N''EXISTE PAS !!');
   1 :WRITE('ERREUR: DESCRIP #',n2:3,' N''EXISTE PAS !!');
   2 :WRITE('ERREUR: TAMPONS DIFFERENTS UTILISES!!');
   3 :WRITE('ERREUR: ANALYSE #',n2:3,' DISP <> BATCH !!');
   4 :WRITE('ERREUR: DILUTIONS DIFFERENTES      !!')
   else ;
   end;
   WRITE (' DEMANDE ANNULEE ...');
end;
```

```
PROCEDURE valid_grp (VAR groupe:t_groupe_ana;nro:INTEGER);

VAR     i                : INTEGER;
        nro_prec         : INTEGER;
        dilu_prec        : INTEGER;

BEGIN
WITH groupe DO
BEGIN
        pos(5+(nro-1)*2,2);     WRITE ('GROUPE #',nro:2,' : ');
        FOR i := 1 TO nbr_ana DO
        BEGIN
        SEEKREAD(filana,tab_ana[i]);
        IF (IORESULT<>0) or (filana^.nro_a <> tab_ana[i])
        THEN BEGIN
                error (0,nr0,tab_ana[i]);
                nbr_ana := 0;
                EXIT;
             END;
        SEEKREAD(fildsc,filana^.nro_descr);
        IF (IORESULT<>0) or (fildsc^.nro_d <> filana^.nro_descr )
        THEN BEGIN
                error (1,nro,filana^.nro_descr);
                nbr_ana := 0;
                EXIT;
             END;
        if not (fildsc^.dispatching in ['b','B'])
        then begin
                error(3,nro,filana^.nro_a);
                exit;
             end;
        tab_p_qte[i] := fildsc^.qte_utilisee;
        if fildsc^.nro1_tampon = maxint
                then tab_t_nro[i] := 0
                else tab_t_nro[i] := fildsc^.nro1_tampon;
        if fildsc^.qte1_tampon = maxint
                then tab_t_qte[i] := 0
                else tab_t_qte[i] := fildsc^.qte1_tampon;
        tab_dbl_rcg[i] := fildsc^.dbl_rincage;
```

```
        IF (i>1) AND (option in ['d','D'])
                 THEN
                 IF (fildsc^.qte1_tampon div fildsc^.qte_utilisee)
                 <>dilu_prec
                          THEN BEGIN
                                   error(4,nro,0);
                                   nbr_ana := 0;
                                   EXIT;
                               END;
         IF (i>1) AND (option in ['t','T','d','D'])
                 THEN IF tab_t_nro[i] <> nro_prec
                          THEN BEGIN
                                   error(2,nro,tab_t_nro[i]);
                                   nbr_ana := 0;
                                   EXIT;
                               END;
         nro_prec := tab_t_nro[i] ;
         dilu_prec := fildsc^.qte1_tampon div fildsc^.qte_utilisee;
         END;
         WRITE ('OK .');
END;
END;

PROCEDURE valider (VAR tab_groupe:t_tab_groupe;VAR nb_groupe:INTEGER);

VAR      i : INTEGER;
         c : CHAR;

BEGIN
         home; cteos;
         cadre1;
         pos (1,18);
         WRITE ('VALIDATION DES ',nb_groupe:2,' GROUPES D''ANALYSE .');
         FOR i := 1 to nb_groupe do
         valid_grp(tab_groupe[i],i);
         pos(21,20);
         bell(2);
         write('Tapez sur une touche pour continuer');
         read(c);
end;

MODEND.
```

12.2 Gestion références et demandes.

12.2.1 Signification des principales variables.

```
t_ref            = RECORD
        nro_ref : str_ref;     référence
        nro_dde : integer;     numéro du secteur où se trouvent les dde
        nro_pos : integer;     position de cette référence sur le rack
     end;

t_tab_ref        = ARRAY [1..max_ref] of t_ref    ;
      table de références
```

12.2.2 Description des fonctions.

PROCEDURE load_ref (VAR t: t_tab_ref; var n: integer);

Cette procédure renvoie dans "t[i].nro_ref" la ième référence du fichier FILREF et renvoie i dans "t[i].nro_dde". Ce i correspond en fait a la fois à la position de la référence dans FILREF et à la position de la demande dans FILDDE.(Cfr 4.2.3) Elle renvoie également dans "n" le nombre de références lues. A chaque référence est donc associé le numéro de la demande correspondante; cela peut sembler ici quelque peut redondant mais l'utilité réside dans le fait que les références vont être triées et si chaque référence garde son numéro de demande, il est inutile de trier également le fichier des demandes puiqu'on y a un accès direct par le numéro. Remarques :

- Seul le fichier FILREF est chargé en mémoire car sa taille est relativement faible. Le fichier FILDDE quant à lui est trop volumineux (au pire 500*64 entiers = 64k) il est donc laissé sur disque mais on s'y réserve toutefois un accès direct.

- "t[i].nro_pos" est indeterminé.

- i commence a 0.

PROCEDURE tri_ref (var t: t_tab_ref; n: integer);
Cette procédure effectue, dans un premier temps, un tri alphabétique de la table "t" de "n" références. Toutes les références commencant par la même lettre vont donc être contigües. Ce tri va donc donner une nouvelle table qui en fait est une suite de sous-tables qui se caractérisent par un premier caractère identique. Dans un deuxième temps, la procédure va assigner à "t[i].nro_pos" la position relative de la référence "t[i].nro_ref" par rapport au premier élément de la sous-table à laquelle elle appartient.
EXEMPLE:

- la 3ème référence commencant par J aura un nro_pos valant 3
- la 25ème référence commencant par Z aura un nro_pos valant 25
- etc...

FUNCTION testdde (dde: ddeanalyse; nroanalyse: INTEGER) : BOOLEAN;
renvoie true si l'analyse nroanalyse est positionnée dans la table dde, renvoie false sinon.

PROCEDURE setdde (VAR dde: ddeanalyse; nroanalyse: INTEGER);
positionne l'analyse nroanalyse dans le tableau dde

PROCEDURE resetdde (VAR dde: ddeanalyse; nroanalyse: INTEGER);
annule l'analyse nroanalyse dans le tableau dde

12.2.3 Texte source.

```
MODULE gesdde;

CONST   max_ref         = 500 ;

TYPE    str_ref         = string[6]                     ;
        ddeanalyse      = ARRAY [0..63] OF INTEGER ;
        t_ref           = RECORD
                                nro_ref : str_ref       ;
                                nro_dde : integer       ;
                                nro_pos : integer       ;
                          end;
        t_tab_ref       = ARRAY [1..max_ref] of t_ref   ;


VAR     filref          : EXTERNAL text;
        fildde          : EXTERNAL FILE OF ddeanalyse   ;

EXTERNAL FUNCTION power(n,exp:INTEGER) : INTEGER;

PROCEDURE load_ref (VAR t:t_tab_ref;var n:integer);

var     i : integer;

begin
        i := 0;
        while (not eof(filref)) and (i<max_ref) do
        begin
                i:=i+1;
                readln(filref,t[i].nro_ref);
                t[i].nro_dde := i-1;
        end;
        n:=i;
end;
```

```
PROCEDURE tri_ref (var t:t_tab_ref;n:integer);

var     itmp,i,j        : integer;
        stmp            : str_ref;
        c_prec          : char;

begin
        for i:=1 to n-1 do
        begin
                for j:=1 to n-i do
                if t[j].nro_ref > t[j+1].nro_ref
                then begin stmp             := t[j].nro_ref;
                           itmp             := t[j].nro_dde;
                           t[j].nro_ref     := t[j+1].nro_ref;
                           t[j].nro_dde     := t[j+1].nro_dde;
                           t[j+1].nro_ref   := stmp;
                           t[j+1].nro_dde   := itmp;
                     end;
        end;
        c_prec := '!';
        for i:= 1 to n do
        begin
                if t[i].nro_ref[1] <> c_prec
                        then j:=1  else j:=j+1;
                c_prec := t[i].nro_ref[1];
                t[i].nro_pos := j;
        end;
end;

FUNCTION testdde (dde:ddeanalyse;nroanalyse:INTEGER) : BOOLEAN;

VAR     nroentier,nrobit : INTEGER ;

BEGIN
        nroentier := nroanalyse div 16  ;
        nrobit    := nroanalyse mod 16   ;
        IF (dde[nroentier] & power(2,nrobit) <> 0)
                THEN testdde := TRUE
                ELSE testdde := FALSE;
END;
```

```
PROCEDURE setdde (VAR dde:ddeanalyse;nroanalyse:INTEGER);

VAR     nroentier,nrobit : INTEGER ;

BEGIN
        nroentier := nroanalyse div 16  ;
        nrobit    := nroanalyse mod 16   ;
        dde[nroentier] := dde[nroentier] ! power(2,nrobit);
END;



PROCEDURE resetdde (VAR dde:ddeanalyse;nroanalyse:INTEGER);

VAR     nroentier,nrobit : INTEGER ;

BEGIN
        nroentier := nroanalyse div 16  ;
        nrobit    := nroanalyse mod 16   ;
        dde[nroentier] := dde[nroentier] & ~( power(2,nrobit));
END;

MODEND.
```

12.3 Dispatching.

12.3.1 Signification des principales variables.

Les principales variables ont déjà été définies dans les modules précédents.

12.3.2 Description des fonctions.

```
PROCEDURE INITIALISATION;
                effectue l'initialisation des  transmission et du
                microlab

PROCEDURE nettoyage (n: integer);
                effectue n nettoyages

PROCEDURE trt_insuff (n: integer);
                traite le cas où le microlab n'a pas trouvé assez
                de liquide dans une éprouvette

PROCEDURE   load_tampon   (var   n:   t_nom_tampon;   var   x,y:
                t_pos_tampon; var nb: integer);
                charge  les noms et les positions des  différents
                tampons du fichier TAMPON

PROCEDURE pos_rack2(nro_gr: integer);
                positionne le bras de l'hamilton au-dessus  d'une
                cupule du rack secondaire.

PROCEDURE next_pos_rack2(nro_gr: integer);
                calcule la prochaine position  libre dans le rack
                secondaire

PROCEDURE trt_groupe (grp: t_groupe_ana; n_gr: integer);
                Cette  procédure sera expliquée plus en détail  à
                la  page  suivante  car  il  s'agit  en  fait  du
                traitement principal du dispatching.
```

ALGORITHME DE TRAITEMENT D'UN GROUPE:

```
pour chaque référence
        lire son tableau des demandes
        pour toutes les analyses du groupe
                tester si cette analyse est demandée
                si oui
                   cumuler les quantités nécessaires
        si l'option est "dilution" ou "prélèvement"
        et si le cumul des qtés de prélèvement est > 0
        alors
                se positionner à l'endroit du rack de gauche
                correspondant à la référence en cours
                pomper la qté cumulée de prélèvement
                et la recracher dans le rack de droite
                à la position voulue
        si l'option est "dilution" ou "tampon"
        et si le cumul des qtés de tampon est > 0
        alors
                se positionner à l'endroit du tampon requis
                pomper la qté cumulée de tampon
                et la recracher dans le rack de droite
                à la position voulue
```

12.3.3 Programme principal.

- Initialisations:
  * de la transmission (initrsm)
  * des pompes
  * du bras
- Ouverture des fichiers.
- Chargement du fichier POSREF; si celui-ci a une mauvaise structure, arrêt du programme.
- Chargement du fichier TAMPON et affichage à l'écran.
- Chargement des références.(load_ref)
- Tri des références.(tri_ref)
- Calcul de la position de chaque référence dans le rack en quinconce:
  * position = position de base + position dans la sous liste commencant par la même lettre - 1

- Lecture à l'écran des différents groupes.(get_groupe)
- Validation de ces groupes.(valider)
- Traitement de chaque groupe.(trt_groupe)
- Fermeture des fichiers.

12.3.4 Utilisation.

- L'écran 1 constitue l'entête du programme.
- L'écran 2 reste affiché pendant la durée des initialisations
- L'écran 3 affiche le contenu du fichier TAMPON
- L'introduction du premier groupe d'analyses peut commencer:
  * écran 4: il faut donner l'option choisie : t (tampon seulement), p (prélèvement seulement), d (tampon + prélèvement) ou s pour stopper.
  * écran 5 : numéro de la première colonne du rack secondaire à utiliser pour le dispatching de ce groupe.
  * écran 6 : nombre de colonnes à utiliser pour ce groupe.
- L'écran 7 rappelle les options choisies et demandes les numéros des analyses à dispatcher dans le groupe #1
- Une fois ces numéros introduits, il demande confirmation avant de passer au groupe #2 (écran 8)
- L'écran 9 est le résultat de l'introduction du groupe #2.
- Au moment d'introduire le groupe #3, on choisi "s" pour stopper et le programme valide alors les deux groupes introduits (écran 10)
- Le programme se poursuit par l'exécution du dispatching pour les groupes "OK"

écran 1

```
DISPATCHING BATCH.

                                        AUTEUR: E.ALEXANDRE

                                Tous droits strictement reserves
```

Tapez sur une touche pour commencer

écran 2

```
Initialisations en cours  .....

Initialisations terminees .....
```

écran 3

```
                DISPOSITION DES TAMPONS

1 : TAMPON1
2 : TAMPON2
3 : TAMPON3
4 : TAMPON4
5 : TAMPON5
6 : TAMPON6
7 : TAMPON7
8 : TAMPON8

                        Tapez sur une touche pour continuer
```

écran 4

```
                GROUPE D'ANALYSE # 1

OPTION ? (t,d,p) ou (s) pour stopper :
```

écran 5

```
GROUPE D'ANALYSE # 1

OPTION ? (t,d,p) ou (s) pour stopper : t
POSITION (entre  1 et 19) :
```

écran 6

```
GROUPE D'ANALYSE # 1

OPTION ? (t,d,p) ou (s) pour stopper : t
POSITION (entre  1 et 19) : 1
NOMBRE DE COLONNES (entre 1 et 19) :
```

écran 7

```
                    GROUPE D'ANALYSE # 1

ANALYSES:

OPTION : TAMPON UNIQUEMENT        POSITION :  1      NBRE DE COLONNES :  3
```

écran 8

```
                    GROUPE D'ANALYSE # 1

ANALYSES:
          52          25
          53          26
          54          27
          55
          89
          124
          541
          14
          12
          13

OPTION : TAMPON UNIQUEMENT        POSITION :  1      NBRE DE COLONNES :  3

                                                  OK  (o/n) ? :
```

écran 9

```
                    GROUPE D'ANALYSE # 2

ANALYSES:
          1

 OPTION : DILUTION                POSITION :  5      NBRE DE COLONNES :  2

                                                    OK  (o/n) ? :
```

écran 10

```
                VALIDATION DES  2 GROUPES D'ANALYSE.

GROUPE # 1 : ERREUR: ANALYSE # 52 N'EXISTE PAS !! DEMANDE ANNULEE ...

GROUPE # 2 : OK .

                                  Tapez sur une touche pour continuer
```

12.3.5 Texte source.

```
PROGRAM dispat;

CONST
        bulle           = 10;
        lmax            = 20;
        max_tampon      = 10;
        qte_rincage     = 100;
        max_ana_simul   =  40;
        max_groupe      =   5;
        max_ref         = 400;

TYPE    t_tab_int       = ARRAY [1..max_ana_simul] OF INTEGER;

        t_tab_bool      = ARRAY [1..max_ana_simul] OF BOOLEAN;

        t_groupe_ana    = RECORD
                            nbr_ana         : INTEGER ;
                            tab_ana         : t_tab_int ;
                            tab_p_qte       : t_tab_int ;
                            tab_t_nro       : t_tab_int ;
                            tab_t_qte       : t_tab_int ;
                            tab_dbl_rcg     : t_tab_bool ;
                            position        : INTEGER ;
                            nb_col          : INTEGER ;
                            option          : CHAR ;
                          END;

        t_tab_groupe    = ARRAY [1..max_groupe] OF t_groupe_ana;
```

```
tab27     = ARRAY [1..27] OF INTEGER

 t_analyse =
         RECORD
                 nro_a              : INTEGER
                 libelle            : STRING[25]
                 nro_descr          : INTEGER
         END;

 t_description =
         RECORD
                 nro_d              : INTEGER
                 prelevement        : CHAR
                 technique          : CHAR
                 dispatching        : CHAR
                 dbl_rincage        : BOOLEAN
                 qte_utilisee       : INTEGER
                 nro1_tampon        : INTEGER
                 qte1_tampon        : INTEGER
                 nro2_tampon        : INTEGER
                 qte2_tampon        : INTEGER
         END;
```

```
str4      = string[4 ]                          ;
 str20             = string[20]                          ;
 str_ref           = string[6]                           ;
 ddeanalyse        = ARRAY [0..63] OF INTEGER            ;
 t_ref             = RECORD
                           nro_ref : str_ref             ;
                           nro_dde : integer             ;
                           nro_pos : integer             ;
                     end;
 t_tab_ref         = ARRAY [1..max_ref] of t_ref         ;
 t_pos_tampon      = ARRAY [1..max_tampon] OF INTEGER;
 t_nom_tampon      = ARRAY [1..max_tampon] OF STRING;
 t_pos_cour        = ARRAY [1..max_groupe] OF INTEGER;
 t_tab_pos_ref     = ARRAY ['A'..'Z'] OF INTEGER;

VAR
 filref            : text                                ;
 posref            : text                                ;
 filana            : FILE OF t_analyse                   ;
 fildsc            : FILE OF t_description               ;
 fildde            : FILE OF ddeanalyse                  ;
 tampon            : text                                ;
 outfile           : FILE OF CHAR                        ;
 i,j,k             : INTEGER                             ;
 tab_ref           : t_tab_ref                           ;
 liste_groupe      : t_tab_groupe                        ;
 nombre_groupe     : INTEGER                             ;
 posx_tampon       : t_pos_tampon                        ;
 posy_tampon       : t_pos_tampon                        ;
 nom_tampon        : t_nom_tampon                        ;
 pos_cour          : t_pos_cour                          ;
 tab_pos_ref       : t_tab_pos_ref                       ;
 c,c1,c2           : CHAR                                ;
 nbr_tampon        : INTEGER                             ;
 tot_prel          : INTEGER                             ;
 tot_tamp          : INTEGER                             ;
 nro_tamp          : INTEGER                             ;
 nbr_rinc          : INTEGER                             ;
 cont_pompe2       : INTEGER                             ;
 nbr_ref           : INTEGER                             ;
 err2              : INTEGER                             ;
 qte               : INTEGER                             ;
 result            : INTEGER                             ;
```

```
{declaration des routines externes
-----------------------------------}
{MODULE SCREEN}
EXTERNAL FUNCTION  OUI          (ST:STRING;C:CHAR):BOOLEAN;
EXTERNAL FUNCTION  INPUTINT     (ST:STRING;C:CHAR):INTEGER;
EXTERNAL PROCEDURE CADRE ;
EXTERNAL PROCEDURE CTEOL;
EXTERNAL PROCEDURE CTEOS;
EXTERNAL PROCEDURE CSCR;
EXTERNAL PROCEDURE HOME;
EXTERNAL PROCEDURE BELL         (N:INTEGER);
EXTERNAL PROCEDURE DESABLE;
EXTERNAL PROCEDURE BLINK;
EXTERNAL PROCEDURE GRAPHIC;
EXTERNAL PROCEDURE REVERSE;
EXTERNAL PROCEDURE POS(L,C:INTEGER);


{MODULE SOFTCOMM}
EXTERNAL PROCEDURE setwopt ;
EXTERNAL PROCEDURE resetwopt;
EXTERNAL PROCEDURE initrsm;
EXTERNAL PROCEDURE pinixyz ;
EXTERNAL PROCEDURE piniz ;
EXTERNAL PROCEDURE princage ;
EXTERNAL PROCEDURE pvidange ;
EXTERNAL PROCEDURE pabs         (x,y,z:INTEGER;      VAR err:INTEGER)
EXTERNAL PROCEDURE pzabs        (zposabs:INTEGER;    VAR err:INTEGER)
EXTERNAL PROCEDURE pzrel        (zposrel:INTEGER;    VAR err:INTEGER)
EXTERNAL PROCEDURE pzdip        (                    VAR err:INTEGER)
EXTERNAL PROCEDURE submerge     (npas:INTEGER;       VAR err:INTEGER)
EXTERNAL PROCEDURE defzpos      (zm,zc,zd:INTEGER;   VAR err:INTEGER)
EXTERNAL PROCEDURE defzvdg      (zpos:INTEGER ;      VAR err:INTEGER)
EXTERNAL PROCEDURE defzrcg      (zpos:INTEGER;       VAR err:INTEGER)
EXTERNAL PROCEDURE defzmvt      (zpos:INTEGER;       VAR err:INTEGER)
EXTERNAL PROCEDURE repmzpos     (VAR mzpos:INTEGER)
EXTERNAL PROCEDURE repxpos      (VAR xpos:INTEGER)
EXTERNAL PROCEDURE repypos      (VAR ypos:INTEGER)
EXTERNAL PROCEDURE repzpos      (VAR zpos:INTEGER)
EXTERNAL PROCEDURE pmpout       (npmp,npas:INTEGER; VAR err:INTEGER)
EXTERNAL PROCEDURE pmpin        (npmp,npas:INTEGER; VAR err:INTEGER)
EXTERNAL PROCEDURE refout       (npmp,npas:INTEGER; VAR err:INTEGER)
EXTERNAL PROCEDURE pmpini       (npmp:INTEGER)
EXTERNAL PROCEDURE pmptest      (npmp:INTEGER;    VAR ready:BOOLEAN)
EXTERNAL PROCEDURE dwait        (nrodev:INTEGER)
```

```
(MODULE EXTDCOMM)
EXTERNAL PROCEDURE trterr (nrodev,err:INTEGER);
EXTERNAL PROCEDURE trtready (nropompe:INTEGER);
EXTERNAL PROCEDURE check;
EXTERNAL PROCEDURE prackg (nrox,nroy,h,hmvt:INTEGER);
EXTERNAL PROCEDURE prackd (nrox,nroy,h,hmvt:INTEGER);
EXTERNAL PROCEDURE pxrel (xposrel:INTEGER;VAR err:INTEGER);
EXTERNAL PROCEDURE pyrel (yposrel:INTEGER;VAR err:INTEGER);
EXTERNAL PROCEDURE prackqc (position,h,hmvt:integer);

(MODULE GESDDE)
EXTERNAL PROCEDURE load_ref (VAR t:t_tab_ref;var n:integer);
EXTERNAL PROCEDURE tri_ref  (var t:t_tab_ref;n:integer);
EXTERNAL FUNCTION  testdde  (    dde:ddeanalyse; nroanalyse:INTEGER)
                 : BOOLEAN;
EXTERNAL PROCEDURE setdde   (VAR dde:ddeanalyse; nroanalyse:INTEGER);
EXTERNAL PROCEDURE resetdde (VAR dde:ddeanalyse; nroanalyse:INTEGER);

(MODULE GESGRP)
EXTERNAL PROCEDURE
        get_groupe (VAR tab_gr:t_tab_groupe;VAR nb_groupe:INTEGER);
EXTERNAL PROCEDURE
        valider    (VAR tab_gr:t_tab_groupe;VAR nb_groupe:INTEGER);
EXTERNAL PROCEDURE cadre1;

EXTERNAL PROCEDURE @hlt;
```

```
PROCEDURE INITIALISATION;

VAR     i       : INTEGER;
        ch      : CHAR;

BEGIN
        home;
        cteos;
        cadre;
        pos(8,24);
        WRITE('DISPATCHING BATCH.');
        pos(15,50);
        WRITE('AUTEUR: E.ALEXANDRE');
        pos(17,30);
        WRITE('Tous droits strictement reserves');
        FOR i :=1 TO MAXINT DO ;
        bell(5);
        pos(23,40);
        WRITE('Tapez sur une touche pour commencer');
        read(ch);
        home;
        cteos;
        reverse;
        WRITE('Initialisations en cours  .....');
        initrsm;
        setwopt;
        check;
        WRITELN;
        WRITELN;
        WRITELN('Initialisations terminees .....');
end;
```

```
PROCEDURE nettoyage (n:integer);

VAR     i : INTEGER ;

begin
        pvidange;
        refout(1,1000,err2);
        resetwopt;
        pzabs(650,err2);
        setwopt;
        pmpin(1,10,err2);
        defzmvt(650,err2);
        princage;
        for i := 1 to n do
        begin
        if cont_pompe2 =0
        then begin
                pmpin(2,1000,err2);
                cont_pompe2 := 1000;
             end;
        refout(2,qte_rincage ,err2);
        cont_pompe2 := cont_pompe2 - qte_rincage;
        end;
        pzabs(650,err2);
end;


PROCEDURE trt_insuff (n:integer);

var     c : char;

begin
        REPEAT
        bell(50);
        reverse;
        writeln;
        writeln('QUANTITE INSUFFISANTE !!');
        writeln('REMPLISSEZ ET TAPEZ SUR UNE TOUCHE POUR RECOMMENCER')
        read (c);
        desable;
        submerge (n,err2);
        until err2 = 0;
end;
```

```
PROCEDURE load_tampon
     (var n:t_nom_tampon;var x,y:t_pos_tampon;var nb:integer);

var      i : integer;

begin
         i:=0;
         while (i<=max_tampon) and (not eof(tampon)) do
         begin
                  i :=i +1;
                  readln(tampon,n[i]);
                  readln(tampon,x[i]);
                  readln(tampon,y[i]);
         end;
         nb := i;
end;


PROCEDURE pos_rack2(nro_gr:integer);

VAR l,c: INTEGER;

begin
         l:=((pos_cour[nro_gr] - 1 ) div 12) + 1;
         c:=((pos_cour[nro_gr] - 1 ) mod 12) + 1;
         prackd(l,c,1450,500);
end;
```

```
PROCEDURE next_pos_rack2(nro_gr:integer);

var c: char;

begin
        pos_cour[nro_gr] := pos_cour[nro_gr]+1;
        if pos_cour[nro_gr] >=
        ((liste_groupe[nro_gr].nb_col * 12)+
        (liste_groupe[nro_gr].position - 1) * 12 +1)
        then begin
                bell(50);
                writeln;
                reverse;
                writeln('LE RACK DE DROITE EST PLEIN !!!!!!');
                writeln('METTEZ-EN UN VIDE OU REMPLACEZ LES CUPULES');
                writeln('CELA FAIT, TAPEZ SUR UNE TOUCHE POUR CONTINU
                desable;
                read(c);
                pos_cour[nro_gr]
                      :=(liste_groupe[nro_gr].position - 1) * 12 +1;
            END;
end;
```

```
PROCEDURE trt_groupe (grp:t_groupe_ana;n_gr:integer);

var     i,j             : integer;
        next,nett       : boolean;

begin
with grp do
begin
        writeln('TRAITEMENT GROUPE #',n_gr:1);
        for i:= 1 to nbr_ref do
        begin
        WRITELN('REFERENCE ',TAB_REF[I].NRO_REF);
        nett := false;
        nbr_rinc :=0;
        tot_prel :=0;
        tot_tamp :=0;
        nro_tamp :=0;
        seekread(fildde,tab_ref[i].nro_dde);
        if ioresult <> 0
         then BEGIN
             writeln('SEEKREAD DEMANDE Nro ',
                  tab_ref[i].nro_dde:3,'/ ERREUR');
             EXIT;
              end;
        for j := 1 to nbr_ana do
        begin
                if testdde(fildde^,tab_ana[j]) then
                begin
                WRITELN('ANALYSE ',TAB_ANA[J]:4,' DEMANDEE');
                tot_prel := tot_prel + tab_p_qte[j];
                tot_tamp := tot_tamp + tab_t_qte[j];
                if (nbr_rinc<=1)
                 then begin
                        if tab_dbl_rcg[j] then nbr_rinc := 2
                                          else nbr_rinc := 1;
                      end;
                nro_tamp := tab_t_nro[j];
                end;
        end;
        WRITELN('TOT PREL: ',TOT_PREL);
        WRITELN('NRO TAMP: ',NRO_TAMP);
        WRITELN('TOT TAMP: ',TOT_TAMP);
        WRITELN('NBR RINC: ',NBR_RINC);
        WRITELN;
        if ((option in ['t','T','d','D']) and (tot_tamp>0))  or
            ((option in ['p','P','d','D']) and (tot_prel>0))
                 then next := true else next := false;
```

```
        while (option in ['t','T','d','D']) and (tot_tamp>0) do
         begin
                 WRITELN('DISTRIBUTION TAMPON');
            pabs(posx_tampon[nro_tamp],posy_tampon[nro_tamp],650,err2);
            if tot_tamp >1000
            then  qte := 1000
            else  qte := tot_tamp;
            tot_tamp := tot_tamp - qte;
            pmpout(1,bulle,err2);
            submerge(100,err2);
            if err2 <> 0
            then trt_insuff(100);
            pmpout(1,qte,err2);
            pos_rack2(n_gr);
            refout(1,qte+bulle,err2);
            nett := true;
         end;
         if nett then begin defzmvt(650,err2); nettoyage(1); end;

         nett := false;
         while (option in ['p','P','d','D']) and (tot_prel>0) do
         begin
                 WRITELN('DISTRIBUTION PRELEVEMENT');
            prackqc(tab_ref[i].nro_pos,650,650);
            IF tot_prel >1000
            then  qte := 1000
            else  qte := tot_prel;
            tot_prel := tot_prel - qte;
            pmpout(1,bulle,err2);
            submerge(100,err2);
            if err2 <> 0
            then trt_insuff(100);
            pmpout(1,qte,err2);
            pos_rack2(n_gr);
            refout(1,qte+bulle,err2);
            nett := true;
         end;
         if nett then begin defzmvt(650,err2); nettoyage(nbr_rinc); end;
         if next then next_pos_rack2 (n_gr);
         end; (for i)
end;
end;
```

```
BEGIN
        initialisation;
        setwopt;
        pinixyz;
        defzpos(1480,650,1000,err2);
        ASSIGN (filref,'b:filref.dat');
        ASSIGN (fildde,'b:fildde.dat');
        ASSIGN (tampon,'b:tampon.dat');
        ASSIGN (fildsc,'b:fildsc.dat');
        ASSIGN (filana,'b:filana.dat');
        ASSIGN (posref,'b:posref.dat');
        RESET(posref);
        RESET(fildsc);
        RESET(filana);
        RESET(tampon);
        RESET(filref);
        RESET(fildde);
        for c1:='A' to 'Z' do
        begin
                readln(posref,c2);
                if c2 <> c1
                then begin
                    bell(20);
                    writeln('MAUVAISE STRUCTURE DU FICHIER "POSREF.DAT"·
                    @hlt;
                     end;
                readln(posref,tab_pos_ref[c1]);
        end;
        load_tampon(nom_tampon,posx_tampon,posy_tampon,nbr_tampon);
        cadre1;
        pos(1,25); write('DISPOSITION DES TAMPONS');
        for i:=1 to nbr_tampon do
        begin
                pos(4+i,10);
                write(I:2,' : ',nom_tampon[i]);
        end;
        bell(1);
        pos (21,20); write('Tapez sur une touche pour continuer ');
        read(c);
```

```
load_ref (tab_ref,nbr_ref);
 tri_ref  (tab_ref,nbr_ref);
 for i:=1 to nbr_ref do
 tab_ref[i].nro_pos  :=   tab_ref[i].nro_pos +
                         tab_pos_ref[tab_ref[i].nro_ref[1]] - 1
 cont_pompe2 := 0;
 get_groupe (liste_groupe,nombre_groupe);
 valider    (liste_groupe,nombre_groupe);
 home; cteos;
 for i := 1 to nombre_groupe
      do pos_cour[i] := (liste_groupe[i].position - 1) * 12 +1;
 for i := 1 to nombre_groupe
    do trt_groupe (liste_groupe[i],i);
 nettoyage(2);
 pinixyz;
 CLOSE(fildde,result);
 CLOSE(filref,result);
 CLOSE(posref,result);
 CLOSE(tampon,result);
 CLOSE(fildsc,result);
 CLOSE(filana,result);

END.
```

Appendix A

Résumé des commandes des devices.

A.1 DEVICE #1/2.

I	Positionne la valve en INPUT.
O	Positionne la valve en OUTPUT.
Pnnnn	Pomper nnnn pas.(0 <= nnnn <= 1000)
Dnnnn	Refouler nnnn pas.(0 <= nnnn <= 1000)
Snn	Definit la vitesse de plongée (0 <= nn <= 15)
Lnn	Descendre avant de s'arrêter (0 <= nn <= 99)
R	Démarre et exécute un command string
Z	Initialise la pompe. Cette commande exécute en fait la séquence: 1. S4L8D1500R 2. IP30R 3. OD10R
RS	Renvoie dans RM$ l'état de la pompe (Y = prête , N = pas prête)

A.2 DEVICE #3.

PI	Position initiale en x,y,z
ZI	Position initiale en z
PA x,y,z	Positionnement absolu en x,y,z
PP n	Positionnement rack primaire
PS n	Positionnement rack secondaire
PC	Positionnnement au point de rincage
PW	Positionnnement au point de vidange
PR n	Positionnnement réactif (1 <= n <= 10)
ZA zzzz	Positionnnement en z-position absolue zzzz
ZR +/-zzzz	Positionnnement en z-position relative +/- zzzz (+ : vers le bas ; - : vers le haut)
ZD	Positionnnement en z-dip
ZX zzz	Descendre jusqu'au liquide et submerger de zzz pas
SA zm,zc,zd	Définit 3 z-positions pour l'adressage absolu
SP zm,zc,zd	Définit 3 z-positions pour le rack primaire
SS zm,zc,zd	Définit 3 z-positions pour le rack secondaire
SR n,zm,zc	Définit 3 z-positions pour les réactifs
SW zzzz	Définit la z-position pour la vidange
SC zzzz	Définit la z-position pour le rincage
SZ zzzz	Définit la z-position durant le prochain mouvement en x,y
RX	Renvoie la position courante en x

RY	Renvoie la position courante en *y*
RZ	Renvoie la position courante en z
RM	Renvoie la z-position maximale assignée à la position courante en x,*y*

A.3 Generalités.

Signification des paramètres zm,zc,zd.

zm	z-position en dessous de laquelle il est interdit de descendre.
zc	z-position à partir de laquelle doit se faire la recherche de liquide.
zd	z-position jusqu'à laquelle on peut chercher du liquide.

Dimension des pas:

X : 0.1524 mm/pas
Y : 0.1270 mm/pas
Z : 0.1000 mm/pas

Valeurs maximales des nombres de pas:

X : 0000 <= nnnn <=5365 pas
Y : 0000 <= nnnn <=2360 pas
Z : 0000 <= nnnn <=1800 pas

Définitions.

output : endroit où se trouve la pointe de l'aiguille.
input : bouteille d'eau déminéralisee.

Index

Table of Contents