MASTER EN SCIENCES INFORMATIQUES

Conception et implémentation d'un didacticiel d'aide à la lecture rapide

Bournonville, Jacques

Award date:
1987

Awarding institution:
Universite de Namur

FACULTES
UNIVERSITAIRES
N.D. DE LA PAIX
NAMUR

INSTITUT D'INFORMATIQUE

CONCEPTION ET IMPLEMENTATION

d'un

DIDACTICIEL

d'aide à

LA LECTURE RAPIDE

Mémoire de fin d'étude
en vue de l'obtention du grade de
LICENCIE ET MAITRE EN INFORMATIQUE

Année académique 1986-1987

Auteur : Jacques Bournonville

Promoteur : Prof. Claude Cherton

RUE GRANDGAGNAGE, 21, B - 5000 NAMUR (BELGIUM)

REMERCIEMENTS

Je tiens beaucoup à remercier mon promoteur, Mr.Cherton, pour l'intérêt qu'il a porté à mon mémoire. Grâce à sa grande disponibilité, son respect pour mon travail et mes opinions , j'ai pu bénéficier de ses nombreux conseils, tout en gardant une autonomie réelle. Son impact aura certainement été important.

Je remercie tout ceux et celles qui, après s'être donné la peine de lire mon mémoire, m'ont donné un avis dont j'ai pu tenir compte.

TABLE DES MATIERES

PRESENTATION DU MEMOIRE

Il est très impressionnant d'entendre parler de ces hommes qui parcourent quatre journaux en vingt minutes. Quel miracle les rend donc capables de tels prodiges ? Suivez-moi! Nous allons essayer de le découvrir.

De multiples tests tendent à montrer qu'il existe plusieurs catégories de lecteurs. La classification part des "Lents" et s'étend jusqu'aux "Prodiges".

Contrairement à ce qu'on pourrait croire intuitivement, la qualité d'une lecture semble s'améliorer avec sa vitesse d'exécution. C'est à dire qu'en général, plus un lecteur est naturellement rapide, mieux il retient le message, le sens à extraire des textes.

L'utilité de lire vite et bien paraît évidente pour les étudiants, hommes d'affaires, cadres, ...etc... , pour tous ceux qui doivent traiter beaucoup d'informations en un minimun de temps.

En laboratoire, des chercheurs ont découvert comment nous lisons. Ces études ont permis de développer des techniques qui permettraient aux lecteurs de doubler leur vitesse , tout en augmentant leurs capacités de rétention d'information.

Sur base de ces recherches, Francois Richaudeau présente sa méthode et propose des exercices dans un livre parut aux Editions MARABOUT :

"Une méthode moderne pour apprendre sans peine

LA LECTURE RAPIDE."

Si je n'ai pas la compétence pour juger la valeur de son travail, il apparaît cependant qu'un livre n'est pas le support idéal pour développer ses techniques. Un didacticiel, c'est à dire le recours à l'informatique et à sa puissance, semblent présenter de réels avantages.

Après avoir étudié la théorie défendue par F.Richaudeau, je dois concevoir un didacticiel d'aide à la lecture rapide. J'implémenterai et si possible améliorerai un maximum de ses méthodes d'apprentissage.

Ce n'est qu'après une expérimentation longue et bien menée que l'on saura si mes hypothèses, mes propositions, sont vraiment pertinentes, et si mon implémentation tient ses promesses.

AVERTISSEMENTS

Il faut savoir que j'écris ce mémoire en tant qu'informaticien. Mon but n'est pas d'offrir aux pédagogues, des idées dans le cadre de rencontres interdisciplinaires, ou d'élaborer de nouvelles méthodes pédagogiques. Je n'ai aucune compétence pour cela.

Mon rôle est d'appliquer mes connaissances de l'informatique et de sa puissance, pour implémenter le livre de F.Richaudeau sur la lecture rapide. Je n'ai donc pas à critiquer la validité de tel ou tel exercice. Je pose l'hypothèse que ce livre est pertinent, et j'essaie d'y apporter un mieux par l'informatique, non pas en traitant le fond mais bien la forme. Au profane que je suis, la théorie défendue paraît honnête en ce sens qu'elle ne réduit pas les réalités de la lecture dans un modèle mécaniste; modèle résumant tout à ce qu'il peut intégrer.

Suivant les conseils de mon promoteur, je n'ai pas essayé de rencontrer un grand nombre de spécialistes de la lecture rapide. Car il existe un risque réel d'avoir autant de points de vue que d'experts différents. Cela aurait pu entraîner une grosse perte de temps.

Ce sera donc surtout à l'utilisation du didacticiel que l'on connaîtra la validité de mes hypothèses de travail.

Remarquons que le livre de F.Richaudeau qui est à la base de ce mémoire, ne prévoyait pas d'implémentation informatique. Je n'y ai pas toujours trouvé les précisions utiles à la formalisation de la théorie. En effet, il y a de réels obstacles au passage d'une théorie numériquement imprécise, plus descriptive que prescriptive, traitant d'un comportement humain, à une application usant de cette informatique qui ne se nourrit que de vertus impersonnelles, numériques, et normatives .

Il m'a donc fallu prendre des décisions pour suppléer aux manques de la théorie. Pour cela j'ai toujours essayé de conserver un souci de bon sens et de correspondance à la philosophie perçue de François Richaudeau.

Mais il me faut dire un mot sur la place de l'enseignement assisté par ordinateur dans les sciences informatiques.

Il n'existe pas de théorie générale sur ce qu'est un bon didacticiel. Toute les recherches à cet égard sont encore prototypiques. Je ne peux donc pas faire reposer mon travail sur des bases formelles solides et normatives, mais sur mon intuition et sur l'expérience des autres. Ceci explique la très maigre bibliographie de ce mémoire.

Il apparaît déjà que le didacticiel parfait est un mythe. Cela signifie qu'il doit sans cesse évoluer et donc être assez souple pour s'adapter aux découvertes faisant suite à son expérimentation.

Un didacticiel est un produit informatique, qu'il est ardu de "faire entrer dans la peau d'un instituteur". Il est lié à des contraintes techniques pas toujours compatibles avec ce qui touche aux sciences humaines.

On peut donc en conclure que le meilleur didacticiel du monde ne peut pas remplacer le professeur, mais peut tout au plus le décharger provisoirement de certaines tâches.

Mais qu'en est-il de la lecture rapide dans le domaine du didacticiel ?

Il existe déjà un logiciel d'aide à la lecture rapide. Je ne l'ai pas vu fonctionner , mais des témoins dignes de foi m'assurent de sa qualité. Cependant, il traite d'autres matières que la lecture rapide. Il est assez cher. Actuellement 12.000 FB environ.
De plus , si les exercices qu'il propose ont mûrement été réfléchis par toute une équipe de spécialistes pendant 7 ans, je ne crois pas qu'il propose un suivi du lecteur au cours des sessions successives.

AMBITIONS

Mon ambition est de proposer un didacticiel fidèle aux thèses de F.Richaudeau, ayant un prix démocratique, et utilisable sur des micros-ordinateurs répandus. J'insisterai sur son caractères motivant, sur sa capacité de permettre au lecteur de se concentrer, et de guider l'étudiant sur base de ses résultats passés. Le didacticiel devra être ergonomique, facile à utiliser et modulable.

Ce didacticiel ne s'addressera pas aux enfants. En effet, la méthode décrite ici suppose une certaine maturité, un bagage intellectuel, un minimun d'entrainement à la lecture que n'ont pas les enfants qui apprennent à lire. L'âge minimum requis se situe aux alentours de 13 ou 14 ans.

GUIDE DE LECTURE

Ce rapport tente d'expliquer le cheminement qui m'a conduit à l'implémentation informatique du livre de François RICHAUDEAU : "Une méthode moderne pour apprendre sans peine LA LECTURE RAPIDE."

Chapitre 1 :

Puisque nous cherchons à améliorer la lecture, je définirai des types de lectures, et les éléments constitutifs de leurs qualités. Après avoir élu le type que nous traiterons, et avoir décrit la difficulté à évaluer objectivement la qualité d'une lecture, j'exposerai la théorie défendue par François Richaudeau qui permet une approche plus précise, plus "scientifique" de la lecture.

Chapitre 2 :

Connaissant cette théorie, nous passerons à une description de ce qu'il faudrait maîtriser pour une application optimale des exercices d'aide à la lecture rapide et à une description des exercices proposés par F.Rchaudeau dans son livre

Chapitre 3 :

Pour décider s'il est utile d'implémenter le livre, il faut peser les avantages et inconvénients de cette informatisation; par conséquent, nous passerons en revue les inconvénients à l'utilisation du livre, ceux à l'utilisation de l'ordinateur, et enfin les avantages que l'on peut tirer de l'ordinateur.

Chapitre 4 :

Ayant considéré qu'il est intéressant d'implémenter le livre, nous fixerons les fonctions principales du didacticiel et nous analyserons les obstacles au passage d'une théorie jouant sur le flou caractéristique des sciences humaines, à sa formalisation plus propice à l'informatique.

Chapitre 5 :

Nous chercherons des solutions aux problèmes soulevés au chapitre précédent. Nous choisirons la méthode la plus simple et la plus efficace pour formaliser les diagnostics que le didacticiel devra poser. Nous prendrons donc des décisions importantes pour la conception du didacticiel.

Chapitre 6 :

Nous décrirons les possibilités générales offertes par le didacticiel, ainsi que les buts, présentations, déroulement, résultats significatifs, diagnostics généraux et particuliers de chaque exercice. Nous préciserons quelles valeurs-clés entrant dans le diagnostic, sont susceptibles d'être modifiées interactivement par un utilisateur.

Conclusion :

Nous terminerons sur les premières impressions que l'on peut tirer après la conception du didacticiel.

Chapitre 1 : THEORIE SUR LA LECTURE

Dans ce chapitre, nous verrons successivement :

- Les lectures et leurs qualités envisagées.
- Le choix du type de lecture adéquat.
- La difficulté d'evaluation d'une lecture.
- La théorie de la lecture défendue par F.Richaudeau.

1.A) DES LECTURES ET DE LEURS QUALITES ENVISAGEES

Avant de choisir le type de lecture sur lequel nous travaillerons, il faut d'abord définir différentes approches possibles d'un texte, ainsi que les éléments constitutifs de leurs qualités.

Nous nous limiterons à la **LECTURE POETIQUE** ou **INTER-TEXTUELLE**, et à la **PROSAIQUE** :

1.A.1) La Lecture de type Poétique ou Inter-Textuelle.

Elle se caractérise par l'attention que porte le lecteur à attacher à chaque mot une émotion pour la lecture poétique, un lien à d'autres textes pour la lecture inter-textuelle.

Dans un poème, l'attention se porte sur les sons, les couleurs, la longueur et la silhouette des mots, sur la forme du texte. Tout peut se lire et se re-lire. Le temps n'a plus d'importance. Le monde peut se confondre dans les nuages. Il n'y a rien d'autre que recherche du ou des plaisirs.

La lecture inter-textuelle est très utilisée par ceux qui font de la recherche en profondeur sur la signification des textes, par exemple dans l'étude de la Bible. Un mot voit sa signification enrichie aussi par sa fréquence d'apparition dans d'autres textes, par l'amalgame de sens déjà découverts et par l'étude de l'environnement de l'auteur ,la connaissance que l'on a de son époque et sa région.

Ainsi l'Abbaye de Maredsous a procédé à un "encodage" de la Bible, qui permet un traitement rapide des textes. Il n'y a plus de difficultés à savoir combien de fois le mot "EAU" apparaît dans les épîtres de Saint-Jean pour le comparer à sa fréquence d'apparition dans l'évangile de Saint-Marc. Connaissant l'importance de l'irrigation des déserts d'Israël, les origines des deux Saints, l'interêt relatif que chacun porte à l'eau, on lui prêtera une nouvelle signification qui améliorera encore notre perception des textes.

Les approches poétiques et inter-textuelles se ressemblent par leur désir d'associer un maximum de sens ou d'émotions. Une bonne lecture de ce type refuse l'empressement et ne se jauge qu'au plaisir que l'on tire à jouer sur les mots ou à rêver.

1.A.2) La lecture de type Prosaïque.

Par opposition avec l'approche précédente, j'insisterai sur le caractère utilitaire et informatif de ce type de lecture. Distinguons la lecture GLOBALE, et la Lecture SELECTIVE.

a) Lecture Globale.

ELLE suppose un parcours rapide et informatif de **TOUS** les mots d'un texte. Il n'est pas souvent nécessaire de revenir en arrière, ni de systématiquement chercher tous les sens possibles d'un terme . Elle s'applique aussi bien aux journaux, revues, et rapports qu'aux romans. En effet, si l'intérêt est la trame ou le suspense d'une histoire, un roman est aussi plaisant à lire rapidement que lentement.

Ainsi la qualité d'une lecture globale c'est à la fois, la rapidité adaptée à la difficulté et la quantité d'informations utiles retenues mais aussi parfois le plaisir retiré.

b) La lecture sélective.

Elle se caractérise par un rapide parcourt du texte et suppose que le lecteur veut s'informer sur un point précis en rentabilisant son temps au maximum. Parcourant la table des matières, on choisit ses chapitres ou articles qu'on lit en passant précipitamment sur les paragraphes inintéressants, et en ralentissant sur les autres. Elle exige que l'on soit capable de repérer facilement les parties importantes d'un texte, grâce à sa structure . De plus elle dépend fortement du BAGAGE INTELLECTUEL DU LECTEUR ET SES CENTRES D'INTERET.

La qualité de ce type de lecture s'évalue seulement à la quantité d'informations acquises en un minimum de temps.

1.B) DU CHOIX DU TYPE DE LECTURE ADEQUAT

La particularité du didacticiel que je vais développer est qu'il n'est pas vraiment fait pour enseigner une matière ou pour informer mais pour donner des nouveaux réflexes, de nouvelles habitudes de lecture.

Ainsi, si le but recherché est d'améliorer la vitesse et la qualité d'une lecture à l'aide d'un didacticiel, on se rend compte qu'on ne pourra pas traiter de tous les types de lecture décrits précédemment. En effet, la vitesse n'est pas signe de qualité dans l'approche poétique et intertextuelle et en quoi des émotions ou le sens d'un mot seraient-il plus optimaux que d'autres !?...

De même, si la vitesse est un facteur essentiel de la lecture sélective, le bagage intellectuel du lecteur, ses centres d'interêts et son implication vis-à-vis du texte y prennent une telle importance qu'il faut beaucoup plus que de bons réflexes pour pratiquer la lecture sélective.

Ainsi nous choisirons de traiter la lecture GLOBALE.

1.C) DE LA DIFFICULTE D'EVALUATION D'UNE LECTURE

AVERTISSEMENT: Le problème décrit ici a été décrit par F.Mansuet dans son livre "LA LANGUE ECRITE". Je ne l'ai donc pas découvert par mes propres observations;

Pour évaluer une lecture, on a tendance à proposer une lecture silencieuse, suivie d'un questionnaire grâce auquel le lecteur exprime ce qu'il a perçu. Bien entendu, selon que la correction sera du fait de tel ou tel professeur, l'évaluation sera positive ou négative. En effet le correcteur ne peut s'empêcher de s'impliquer dans son travail. Sa correction sera révélatrice de ses propres opinions. Il ne juge pas sur base d'un modèle de lecture, mais sur l'idée qu'il se fait sur le modèle optimal du lecteur. IL NE CHERCHE PAS A SAVOIR CE QUE LE LECTEUR A LU, MAIS A CONNAITRE SON AVIS SUR CE QU'IL A LU.

Ainsi, tant qu'on juge des valeurs, plus rien ne semble objectif. Dans ces conditions, il n'y a pas moyen d'utiliser une unité standard capable d'évaluer le niveau de tout lecteur.

CONSEQUENCES :

On conviendra donc dans ce travail, de ne s'intéresser qu'au contenu du texte en tant qu'information qu'il apporte, et non pas en tant que valeur qu'il défend.

REMARQUE :

Il serait illusoire de croire qu'un questionnaire portant sur un texte puisse être parfaitement objectif. Consciemment ou pas, l'interrogateur portera ses questions sur des points qui lui paraîtront essentiels, et auxquels le lecteurs qui n'a pas les mêmes centres d'intérêt, n'aura porté aucune attention. Il y a donc place à un certain arbitraire.

1.D) THEORIE DE LA LECTURE DEFENDUE PAR F.RICHAUDEAU

Avertissement :

Je répète que n'ayant pas compétence en matière de lecture rapide, je me contente d'exposer cette théorie sans avoir à la critiquer.

La théorie développée ici, n'est pas le fait d'un seul groupe de chercheurs. Ignorant la chronologie des recherches, j'ai tenté de formaliser en tissant un lien logique entre les différents éléments connus. Cela devrait en simplifier la compréhension.

1.D.1) Buts recherchés

On cherche à obtenir :

- un test objectif évaluant la qualité de la lecture.
- Une unité universelle pour l'évaluation du temps de lecture.
- Un modèle décrivant notre façon de lire et expliquant pourquoi certains lisent plus vite et mieux que d'autres.

1.D.2) Le test objectif

Il se compose :

- d' une lecture silencieuse,
- d'un questionnaire sur le contenu informatif pour lequel le lecteur a le choix entre plusieurs solutions proposées. Ces questions ne portent jamais sur un jugement de valeur.

ex) Si le texte traite du cancer, une question ne doit pas être :

"A la lecture de ce rapport on remarque que son auteur est

A) incompétent.

B) mal renseigné.

C) très expert.

"

La question devrait plutôt être de ce genre-ci:

"Les causes du cancer du poumon décrites dans ce texte sont

A) L'abus de tabac.

B) Un régime alimentaire mal équilibré.

C) Mal connues car trop complexes.

"

Si le lecteur répond correctement à au moins 7 questions sur 10, on admettra qu'il a retenu suffisamment de sa lecture. Avec moins de 5 bonnes réponses, le niveau est vraiment faible.

Remarquons que taux de bonnes réponses sera influencé par la connaissance antérieure que le lecteur a du sujet traité dans le texte.

1.D.3) L'unité universelle de vitesse de lecture

En chronomètrant, on peut connaître le temps de lecture d'un texte en secondes . Telle quelle, cette mesure est inutile. En effet, comment la comparer avec celle d'un texte plus long ou plus court traitant d'un autre sujet.

Il faut donc une mesure indépendante de la qualité et de la longueur du texte, mais représentatif d'un temps nécessaire à une lecture valable.

Seront jugés vraiment significatifs, les seuls temps des lectures ayant donné un taux de bonnes réponses supérieur ou égal à 7 sur 10

Pour échapper au problème des longueurs différentes des textes, on n'utilisera pas une mesure absolue en secondes, mais une mesure relative en **NOMBRE DE SIGNES LUS à l'heure**.

Le **SIGNE** étant tout caractère non-blanc du texte.
On a préféré le signe au mot car le temps de lecture d'un mot dépend parfois de sa longueur, et surtout de la connaissance qu'on en a.(On lit plus facilement "papa", que "synallagmatique").

Ainsi, nous possédons une mesure assez précise et objective tant de la vitesse que de la qualité de lecture. Cela permet une approche plus scientifique.

REMARQUE : Pour le sens commun, il est difficile de se représenter ce que signifie concrètement la capacité de lire tel nombre de signes en une heure . Alors pour se donner un repère, on comparera ses propres performances à celles d'autres personnes.

1.D.4) Observations de départ

Grâce au test décrit ci-dessus, on a observé une population hétérogène de lecteurs

Pour un taux de réponse acceptable, une vitesse moyenne est de 113.000 signes/heure. Un très bon lecteur peut aller jusqu'à environ 451.000 signes/heure, et les mauvais descendre à 50.000 signes/heure.

En général, les lecteurs naturellement plus rapides, avaient aussi le meilleur taux de bonne réponse lors du test objectif. On aurait pu se contenter d'expliquer ce bon taux de réponse par le fait qu' arrivé à la fin d'une page, le début est encore frais en mémoire et qu'on intègre donc plus facilement le sens du texte sans sauter en arrière.

1.D.5) Parti pris pour expliquer la lecture

Des chercheurs pensèrent à établir une corrélation entre le mouvement des yeux et la vitesse de lecture.

Pour observer cela, ils filmèrent des yeux puis les aggrandissant à la projection, ils analysèrent leurs déplacements et en déduisirent le parcours.

1.D.6) Comment lisons-nous ?

OBSERVATIONS CHEZ L'ADULTE :

Les yeux se déplacent, se fixent au milieu d'un groupe de mots, puis se déplacent à nouveau. Arrivés en fin de ligne, ils se positionnent rapidement au début de la ligne suivante.

Nous appelerons **POINTS DE FIXATION (P.F.)**, les endroits où les yeux se fixent.

Les mouvements des yeux d'un bon lecteur sont réguliers et les P.F. sont espacés de quelques mots.

Pour les prodiges, la régularité est conservée mais les P.F. peuvent être éloignés de plusieurs lignes.

Chez le mauvais lecteur, le mouvement est saccadé et entaché de fréquents sauts en arrière et/ou il y a un point de fixation par mot.

D'OU ON DEDUIT QUE :

le lecteur reconnaît les mots à la forme lorsque ses yeux sont fixes. S'il est habile, il peut reconnaître un groupes de mots d'un seul coup, et en intégrer le sens alors qu'il déplace ses yeux vers le groupe de mots suivant.

INTEGRER un bout de phrase ou un mot, c'est analyser leur signification par rapport à la phrase dont ils sont issus; c'est comprendre

Un lecteur prodige peut reconnaître 7 lignes d'un coup.

La largeur maximale du groupe de mots que le lecteur reconnaît à la forme, s'appelle le **CHAMP DE FIXATION** ou de **VISION (C.F. ou C.V.)** . (Après échauffement, il est en moyenne de 15 à 20 signes, soit 4 à 5 mots, pour un lecteur habile.)

Le temps moyen pendant lequel les yeux se figent s'appellent **LE TEMPS MOYEN DE FIXATION (TMF)** . Il est de l'ordre de 250 millisecondes. Le lecteur est plus ou moins rapide selon sa position dans le texte. En effet, le temps de lecture est plus long en début et fin de phrase ou de page. Le temps de déplacement des yeux est de l'ordre de 25 millisecondes.

Le temps pour reconnaître un groupe de mots est très proche du temps d'identification d'un seul mot.

Le cerveau peut intégrer plus de mots que notre oeil n'en perçoit Et nos yeux perçoivent plus que nous n'en sommes conscients. Avec l'habitude de lire un peu en avant, notre cerveau peut comprendre ces mots qui nous paraissent flous aux extrémités de notre C.F.

IL EST ESSENTIEL D'ADAPTER SA VITESSE A LA DIFFICULTE DU TEXTE.

1.D.7) Facteurs retenus intervenant dans la vitesse de lecture

Donc, pour lire plus vite, il faut :

- Reconnaitre les mots à la forme.
- Lire par points de fixation.
- Intégrer le sens pendant le mouvement des yeux.
- Adopter un mouvement régulier des yeux.
- Elargir le C.F.

Les défauts qui ralentissent, sont :

- Un C.F. étroit, et ne pas oser lire en avant.
- Ne passer au mot suivant, que lorsque le précédent est intégré.
- Lire mot à mot, ou par mouvements irréguliers.
- Tendances à subvocaliser, c-à-d réciter ce qu'on lit.

Ces sur bases de ces conclusions que F.Richaudeau a développé les excercices décrits plus loin.

REMARQUES IMPORTANTES.

1) Bien sûr d'autres éléments essentiels entrent en compte, mais ils sont d'un aspect plus qualitatif et personnel, difficile à formaliser. Je citerai la mémoire, le bagage culturel, la concentration, l'intèrêt, le vocabulaire, la maîtrise de la langue, l'accord avec les idées de l'auteur.

2) Le TMF et CF ne sont pas des mesures très précises. Le CF est significatif à quelques signes près.

ON A CALCULE QUE LE TMF NE VARIAIT PAS DE PLUS DE 35%.

On ne peut juger de ces valeurs que sur base d'un intervalle de confiance.

Chapitre 2 : DES EXERCICES POUR AMELIORER LA LECTURE

Dans ce chapitre, nous verrons succesivement :

- Les pouvoirs des exercices idéaux.
- Les exercices proposés par F.Richaudeau.

2.A) DES POUVOIRS DES EXERCICES IDEAUX

Si nous supposons que la théorie décrite précédemment est vraie, les exercices augmentant la vitesse et la qualité de lecture devraient posséder certaine qualités dans un environnement adéquat.

- Le lecteur doit comprendre, et être conscient de l'utilité des exercices qu'il effectue.

- Le lecteur doit acquérir de nouveaux réflexes, de nouvelles habitudes. Il doit les utiliser dans sa vie de tous les jours, pour confronter ses performances dans les exercices à la lecture réelle.

- Les exercices ne doivent pas être trop artificiels, et rendre des résultats réellement significatifs du niveau du lecteur, du progrès accomplis, des exercices à effectuer pour s'améliorer.

- Ils doivent être simples, variés, et le moins fatigants possible.

- Ils doivent être personnalisés et adapter la difficulté au niveau du lecteur.

- Ils doivent être présentés de la façon la moins troublante possible.

- Leur contenu doit être fréquemment changé pour ne pas les fausser.

- La présence d'un expert en lecture rapide, capable d'évaluer les résultats de l'étudiant, de le conseiller, et de le stimuler, serait très appréciable.

- Le lecteur doit apprendre à travailler régulièrement.

2.B) EXERCICES PROPOSES PAR F.RICHAUDEAU

Pour ne pas forcer le lecteur à parcourir l'ouvrage de F.Richaudeau, je décrirai ici les exercices qu'il propose.

Vocabulaire : On parlera de **GAMME**, pour les exercices qui doivent être exécutés très fréquemment.

2.B.1) Lecture en colonne

Ceci est un excercice de gamme.

Buts

- Améliorer la capacité à reconnaitre les mots à la forme.
- Elargir le champ de fixation.

Instruments

- Un cache en carton servant à isoler un mot dans une liste.
- Un chronomètre.
- Une liste en colonnes de mots ou de phrases ne dépassant pas 30 signes. La largeur des mots va croissant. Ces mots ne doivent pas avoir de liens entre eux. Le mot suivant doit être imprévisible.

Il faut éviter les séries telles que :

"Do, re, mi, fa, sol, la, si" ou

"nos, vos, leurs, notre, votre, leurs"

Déroulement

- Déclencher le chrono .
- En s'aidant du cache et en fixant le milieu des mots, parcourir la liste aussi vite que possible, en passant au mot suivant dés que le précédent est reconnu. ADOPTER UN RYTHME REGULIER.
- Arrêter le chrono.

Diagnostic

- Un diminution du temps de l'excercice est révélateur d'un élargissement du CF. On s'améliorera en essayant avec des listes de mots plus larges en moyenne.

Inconvénients

- on ne sait pas si le lecteur a réellement reconnu les mots, s' il a été trop vite ou trop lentement pour ses capacités.

- L'excercice est de gamme et quand on a répété 7 fois l'excercice avec les même mots dans le même ordre, on finit par connaître la liste presque par coeur, et par prévoir le mot suivant. Cela fausse l'exercice.

- La largeur des mots n'est pas adaptée au champ de fixation de chacun.

2.B.2) Lecture par points de fixation

Ceci est un excercice de gamme.

Buts

- Donner l'habitude de lire par POINTS de FIXATION, en intégrant le sens des demi-phrases pendant le mouvement des yeux
- Elargir le champ de fixation.

Instruments

- Un chronomètre.
- Un Texte jugé relativement facile à lire pour tous.Il doit être disposé en Y. Les moitiés de lignes se rapprochent l'une de l'autre jusquà former une ligne normale. Puis à la page suivante, on recommence cet écartement, mais avec des demi-lignes um peu plus longues.

EX)

Il essayait ainsi de prévoir
les facultés d'un individu
en examinant les ‹‹bosses››
de son crâne et appelait
cela la phrénologie
On remarque encore aujourd'hui
l'influence des idées de Gall
puisqu'on parle couramment
d'élèves qui ont la
‹‹bosse des mathématiques ››.
Il serait intèressant
de savoir pourquoi
la bosse des mathématiques
etc... etc...

Déroulement

- Déclencher le chrono .
- Prendre le rythme de lecture en 2 points de fixation, et le conserver lorsque les lignes se sont rejointes.
- ADOPTER UN RYTHME REGULIER.
- Arrêter le chrono.

Remarque

- Il va de soi que le lecteur doit adapter sa vitesse à sa position dans le texte, ainsi qu'à la difficulté relative. Ce qui signifie que le temps qu'il s'arrêtera à chaque point de fixation sera variable. Cependant, après avoir lu le texte une fois ou deux, on connaît déjà le sens du texte, on n'a donc plus à réfléchir autant sur la signification. Ainsi le temps de fixation finira par s'égaliser pour chaque point. Ayant acquis une certaine assurance vis-à-vis du texte, ON OSE LIRE QUELQUES MOTS EN AVANT, plus rapidement qu'à la vitesse confortable. Bien sûr, on finit par changer de texte car à lire toujours le même, on finirait par le connaître par coeur, et l'excercice serait faussé.

Diagnostic

- Une diminution du temps peut être un signe qu'on prend l'habitude de lire par points de fixation, avec moins de retour en arrière.

- Dés que la largeur d'une demi-phrase dépasse le CF, le lecteur la lira en deux fois. Donc le temps nécessaire pour la lire sera égal au TMF multiplié par deux. Donc, une petite variation de la longueur d'une phrase peut conduire à une hausse sensible d'un temps de lecture.

Ainsi si le temps de lecture des phrases longues se rapproche de celui des phrases courtes, le CF du lecteur a sans doute grandi.

Inconvénients

- On ne sait pas si le lecteur a réellement compris les phrases, s'il a été trop vite ou trop lentement pour ses capacités.

- Il est arbitraire de décider à la place du lecteur que tel ou tel texte est facile. C'est au lecteur de choisir ses textes en fonction de la difficulté.

- La largeur demi-phrases n'est pas adaptée au champ de fixation de chacun.

REMARQUE : Il est bon que le lecteur effectue une gymnastique qui assouplit les muscles des yeux. En ne lisant que les premiers et derniers mots de chaque phrase d'un texte, le lecteur habitue ses yeux a se déplacer vite et souplement.

2.B.3) Reconnaissance de mots au contour

Ceci est un excercice de gamme.

Buts

- Augmenter les capacités à repérer un mot donné d'après son contour seulement.

Instruments

- Un chronomètre.
- Une liste de 50 ensembles de lignes. La première contient le modèle de mot à retrouver 2 fois dans le reste de l'ensemble.

Exemple)

devenir

arrêter, demeurer, reposer, élever, souffrir, rentrer, devenir, devoir, aller, revenir, retourner, devenir, partir.

Déroulement

- Déclencher le chrono .
- Reconnaître les 50 modèles en un minimun de temps.
- Arrêter le chrono.

Diagnostic

- Il faut arriver à faire l'excercice en moins de 2 minutes 60 , et descendre à 2 minutes. A 2 minutes, on a une bonne capacité à reconnaître les mots à leur contour.

Inconvénients

- On ne sait pas si le lecteur a réellement perçu les mots, s' il a été trop vite ou trop lentement pour ses capacités.

2.B.4) Eviter les confusions de mots

Buts

- Augmenter les capacités à reconnaître un mot mélangé à d'autres qui lui ressemblent.

Instruments

- Un chronomètre.
- Un crayon.
- Une liste de 140 phrases n'ayant pas vraiment de sens, mais comportant des mots semblables. Les phrases ne comportent ni point final, ni majucule de début de phrase.

Exemple)

ta veste est trop vaste
bourre ta bourse de billets neufs !
il mâche son foin avec soin

Déroulement

- Déclencher le chrono .
- Lire les 140 phrases en un minimun de temps.
- Cocher les phrases pour lesquelles on a eu tendance à revenir en arrière.
- Arrêter le chrono.

Diagnostic

- Il faut arriver à faire l'excercice en moins de 2 minutes. Alors si on a fait moins de 4 retours en arrières, l'oeil fait bien la distinction entre les mots qui se ressemblent.

Pour 4 à 8 retours, il faut tendre son attention quand on lit vite pour améliorer le discernement.

Pour plus de 8 retours, il faut s'entraîner à refaire l'excercice.

Inconvénients

- Il y a trop de manipulations : Lancer le chrono, tenir le livre, cocher avec le crayon, tourner les pages, arrêter le chrono. C'est stressant et empêche de se concentrer uniquement sur la lecture.

- Que doit-on faire exactement quand l'excercice dure plus de 2 minutes ?

2.B.5)Tendances à lire mot à mot

Buts

- Mettre en relief les tendances à lire mot à mot.

Instruments

- Un chronomètre.
- Un texte simple d'une vingtaine de lignes.
- Le même texte mais présenté en colonne.

EX) Le texte.

" Cette neige blanche, comme une ouate froide qu'un vent d'hiver avait glacé."

EX) En colonne

Cette	ouate	d'
neige	froide	hiver
blanche,	qu'	avait
comme	un	glacé.
une	vent	

Déroulement

- Déclencher le chrono .
- Lire le texte en colonne le plus vite possible.
- Arrêter le chrono.
- Déclencher le chrono .
- Lire le texte normal.
- Arrêter le chrono.

Diagnostic

- Si on lit le texte en colonne à une vitesse égale ou seulement un peu plus petite à celle du texte normal, c'est qu'on a tendance à lire mot à mot.

Conseil

Effectuer beaucoup d'exercices de lecture par PF.

Inconvénients

- Quelle différence de temps entre les deux lectures est vraiment significative ?

2.B.6)Tendances à subvocaliser

Buts

- Faire prendre conscience au lecteur qu'il a tendance à subvocaliser.

- Faire perdre cette habitude.

Instruments

- Un mouchoir.

- Une liste de phrases dont la lecture orale produit des sons vibratoires.

EX)

Quatre cartes écornées furent sa perte
Ce vin vermeil m'émerveille
Le riz tenta le rat; le rat tenté tâta le riz

Déroulement

- Lire les phrases de plus en plus vite, en tenant le pouce sur le larynx ou en callant un mouchoir au bout des lèvres.

Diagnostic

- Si on sent des vibrations du larynx ou si le mouchoir tombe, on a des tendances à subvocaliser;

Conseil

Il faut refaire l'exercice tant que cette habitude n'est pas perdue

Inconvénients

- Le lecteur est seul à décider s' il a des tendances à subvocaliser.

2.B.7) Lecture naturelle

Buts

- Posséder une évaluation standard de la vitesse de lecture.

- Permettre de tester les vraies améliorations apportées par les exercices.

Instruments

- Un chronomètre.

- Un texte dont on connaît le nombre de signes.

- Un questionnaire sur son contenu informatif. (Voir infra, "théorie de la lecture défendue par F. Richaudeau" au paragraphe "test objectif".)

Déroulement

- Déclencher le chrono .

- Lire le texte le plus vite possible.

- Arrêter le chrono.

- Répondre au questionnaire.

Diagnostic

- Connaissant le temps et la quantité de signes, on calcule la vitesse en NOMBRE DE SIGNES LUS à L'HEURE.

On cacule le taux de bonnes réponses. Pour au moins 7 réponses correctes sur 10, le résultat est très satisfaisant; Pour moins de 5, vous avez lu trop vite ou beaucoup trop lentement, à moins que vous n'ayez pas été assez concentré.

On peut comparer ses résultats à ceux de meileurs ou plus mauvais lecteurs grâce à une échelle allant de 451.000 signes/heure à 50.000signes/heure. C'est plus pratique pour se représenter ses performances intuitivement.

Conseil

- Recommencez l'exercice avec un autre texte, faites la moyenne des résultats. Cela les affinera .

- En s'entraînant à la lecture rapide on peut **DOUBLER** la rapidité, et améliorer le taux de bonnes réponses de 10% à 20%.

Inconvénients

- Quand on ne le connaît pas il est fastidieux de calculer le nombre de signes lus.

CHAPITRE 3 : POURQUOI IMPLEMENTER LE LIVRE ?

Dans ce chapitre, nous verrons succesivement :

- Les inconvenients du livre.
- Les inconvenients de l'ordinateur par rapport au livre.
- Les avantages de l'ordinateur par rapport au livre.

Nous rendant compte des défauts du livre, il faut peser le pour et le contre, avant de décider l'implémentation. Car, si l'informatique a de sérieux atouts, il ne faut jamais oublier de prendre en compte ses limites, pour juger de la pertinence de son utilisation.

3.A) INCONVENIENTS DU LIVRE

- Il faut trop souvent tenir le livre , le chrono et le cache. Ces manipulations sont gênantes, et peuvent nuire à la validité des résultats.

- L'utilisation du livre se révèle fatigante. (Calculer son temps, le noter, comparer.)

- Il n'y a pas d'adaption de la difficulté de l'excercice au niveau du lecteur, et le contenu des exercices ne change pas. (Toujours la même liste de mots en colonne.). C'est démotivant.

- En l'absence de professeur, le lecteur peut avoir du mal à analyser des résultats, et ne se sent pas guidé.

- Le lecteur ne se rend pas toujours compte qu'il a été trop vite ou trop lentement par rapport à ses capacités.

- Il n'est pas toujours conscient de ses progrès.

EN BREF, LE LIVRE EST PEU MOTIVANT, ASSEZ CONTRAIGNANT ET D'UNE UTILISATION PLUTOT RIGIDE. CELA GENE LA CONCENTRATION. AINSI NAQUIT L'IDEE D'UTILISER UN ORDINATEUR POUR AMELIORER L'APPRENTISSAGE.

3.B) INCONVENIENTS DE L'ORDINATEUR PAR RAPPORT AU LIVRE

- Quand on n'en possède pas encore, un ordinateur et ses logiciels peuvent se révéler chers à l'achat. Remarquons cependant que les coûts baissent rapidement.

- A l'exception des "micros" portables dans une valisette, la plupart des micros-ordinateurs exigent la proximité d'une prise de courant avec prise de terre, occupent un plus grand espace de travail et ne sont pas faciles à transporter. On a donc plus de peine à trouver un endroit pour travailler au calme et à son aise.

- A moins de posséder un écran de grande qualité , le clignotement devient pénible pour les yeux

- L'adulte peut ne pas avoir l'occasion de découvrir un didacticiel car il souffre de la " peur de l'ordinateur"; heureusement cette tendance diminue, et peut s'estomper si on développe une bonne ergonomie.

- Un texte se lit plus facilement dans un livre qu'à l'écran.

3.C) AVANTAGES DE L'ORDINATEUR PAR RAPPORT AU LIVRE

Au niveau psychologique.

- Le lecteur qui craindrait la présentation austère d'un livre, peut être motivé par le coté ludique de l'ordinateur.

- Quand le didacticiel évalue les performances du lecteur, celui-ci ne se sent pas jugé. Ses échecs sont instructifs, et il n'a pas à soutenir le regard de son professeur ou de ses condisciples.

- Le lecteur peut être pris en charge par l'ordinateur, qui lui explique la meilleure marche à suivre. Il suivra des conseils ayant des bases objectives qui n'ont rien à voir avec des compatibilités de caractères. Il peut même ne pas tenir compte des conseils sans crainte de blesser des suceptibilités.

- L'ordinateur peut féliciter le lecteur, et s'adapter à son niveau. Il est motivant d'avoir l'impression d'être "compris" (même si ce n'est pas le cas), et de ne pas faire du travail inutile.

Au niveau de la puissance de calcul.

- Calcul automatique du nombre de signes dans un textes.

- Calcul automatique des performances du lecteur.

- Adaptation automatique des excercices aux performances du lecteur. Personnification dans la présentation des résultats.

- Possibilité de simulation d'une lecture, par exemple en mettant en relief les groupes de mots à lire par points de fixation.

- Représentation graphique des résultats.

- Modification aisée du contenu des exercices.

- Capacité de traitement de texte;

Exemples)

a) Générer automatiquement à partir d'un fichier texte, un texte propre à la lecture en 2 points de fixation.

b) Si on a utilisé un texte lors d'un cours, on peut en tirer des mots carctéristiques, pour les excercices. Par exemple, prendre des mots typiquement juridiques dans un texte de droit.

c) Entrer, modifier, ou supprimer facilement les mots et phrases des excercices.

...etc...etc...

Au niveau des manipulations

- Simplifications des manipulations.

Cette analyse comparative des avantages et inconvénients du livre et de l'ordinateur laisse apparaître que la méthode défendue par F.Richaudeau gagnerait à échanger son support livresque contre un support informatique. Nous décidons donc de l'implémenter.

CHAPITRE 4 : DU PASSAGE D'UNE THEORIE A SON IMPLEMENTATION

Dans ce chapitre, nous verrons succesivement :

- Ce que doit faire le didacticiel.
- Les incompatibilités entre l'Homme et la Machine.
- De la difficulté à implémenter la théorie.

4.A) DE CE QUE DOIT FAIRE LE DIDACTICIEL

C'est ici en fait que débute réellement ce mémoire. Nous avons une théorie, des exercices, et un outil informatique; il reste à définir ce qui doit être fait afin d'implémenter un didacticiel d'aide à la lecture rapide.

Schématiquement, les fonctions seront :

- Amélioration de la présentation en simplifiant les manipulations.
- Calculer les résultats , les mémoriser, les présenter de façon personnalisée.
- Réagir aux erreurs du lecteur.
- Aider le lecteur à analyser ses performances.

Dés le début de l'analyse, nous remarquons 2 points importants. Le support informatique et le support livresques sont de nature assez différente pour que le passage de l'un à l'autre ne soit pas un problème trivial. De plus, F.Richaudeau n'ayant pas prévu l'implémentation de son livre, on remarque que la théorie manque d'informations qui nous auraient aidés pour l'informatisation.

4.B) DES INCOMPATIBILITES ENTRE LA MACHINE ET L'HOMME

Je tiens à insister sur le fait qu'il m'a fallu tout un long cheminement et les conseils de mon promoteur pour abandonner quelques néfastes habitudes d'informaticien. En effet, l'habitude de la programmation nous incline à vouloir définir entièrement toutes les éventualités que les algorithmes doivent calculer. Pour cela , nous réduisons l'environnement, et à grands coup d'hypothèses et de pré-conditions, nous contraignons les objets à traiter . En d'autres mots nous ramenons tous à ce que nous pouvons formaliser.

Mais dans quelle mesure peut-on définir un être humain ? Philosophiquement, je refuse de croire qu'on peut réduire l'homme à une équation, et qu'un système expert pouvant modifier ses connaissances, ses règles d'inférence et expliquer ses raisonnements, est capable de posséder une vue suffisemment juste de la réalité humaine.

C'est pourquoi, le système développé sera un système d'AIDE, pas un système expert. je tenterai de limiter les contraintes au maximum, en restant souple pour adapter le logiciel au lecteur et pas l'inverse.

Dans un compromis entre la volonté d'aider le lecteur sans trop l'assujettir à ma vue des choses et celle de formaliser la perception de ses capacités, je ferai appel à sa bonne volonté, à son désir d'apprendre. Face à un sabotage continu ou des erreurs persistantes, le didacticiel sera impuissant. C'est le prix à payer pour l'adaptation de la Machine à l'Homme.

4.C) DE LA DIFFICULTE A IMPLEMENTER LA METHODE

Remarquons la modification de la situation du lecteur par rapport aux exercices. Avec le livre, chacun s'auto-évalue, intuitivement, se rend compte de ses limites et réagit en conséquence. Toutes les responsabilités incombent au lecteur. Alors qu'un didacticiel décharge l'utilisateur de certaines manipulations et le conseille plus systématiquement que dans le livre. Cela signifie que l'algorithme doit percevoir les progrès, et les écarts du lecteur. Il faut donc pour l'ordinateur, définir de façon plus rigoureuse ce que, jusqu'ici, chacun traitait soi-même plus ou moins intuitivement. On se rend compte très vite des imprécisions numériques ou décisionnelles de la théorie défendue par F.Richaudeau, et du problème soulevé pour un contrôle fiable des performances. Cela me force à prendre des décisions pour faire face à ces lacunes.

4.C.1) Imprécisions numériques

1) La règle des 35%.

Prenons l'exemple des lectures en colonne ou par points de fixation. F.Richaudeau écrit qu'on diagnostique une non-reconnaissance des mots en un point de fixation, et un CF trop petit, si le temps de lecture de mots larges est plus long que le temps pour les mots un peu plus courts. Que signifie "plus long" ? Pour résoudre cela, il nous faut définir un intervalle de confiance, en dehors duquel on considèrera que le lecteur a lu trop vite, ou en plus d'un point de fixation . Nous définirons cette intervalle grâce à la règle des 35%.

F.Richaudeau a écrit :" D'un test à l'autre, il y a des écarts maximaux de 35% dans la durée moyenne de fixation."
Malheureusement, F.Richaudeau ne précise pas 35% de QUOI.

Plus loin, je donne mon interprètation de cette règle.

2) Les temps absurdes.

Si certains temps sont significatifs d'une faiblesse du lecteur, d'autres sont représentatifs d'un sabotage ou d'une erreur grave. Par exemple, un TMF de 10 millisecondes est ridiculement petit. Il n'est pas précisé quelle est la limite à partir de laquelle un temps est absurdemment trop long ou trop court.

4.C.2) Imprécisions décisionnelles

1) D'abord F.Richaudeau ne nous dit pas comment analyser et comparer les résultats de différents types d'exercices.

2) Si nous prenons l'exemple du test d'identification de mots semblables, pour lequel on calcule le nombre de sauts en arrières. Si on a mit moins de 2 minutes pour l'exercice, Richaudeau dit comment réagir face aux nombre de ces régressions.

MAIS que doit-on faire et penser quand le lecteur dépasse 2 minutes ?

4.C.3) Problème d'un contrôle fiable

Pour appliquer la règle des 35%, il faut se baser sur une valeur étalon, calculée à un moment pour lequel on est sûr que le lecteur a travaillé correctement. Le gros problème est d'établir cet étalon. Pour cela j'ai envisagé 2 méthodes.

1) Recomposition de mot.

Dans un réflexe d'informaticien méfiant envers les utilisateurs et qui veut contrôler tous ses paramètres, je pensais demander au lecteur de recomposer au clavier le mot qu'il viendrait de lire. Bien sûr, c'est le test idéal pour s'assurer de la reconnaissance des mots, mais cela nuit beaucoup à la pédagogie enseignée. En effet, il est important d'adopter un rythme régulier. Une recomposition tuerait cette régularité, et l'effet recherché serait perdu.

2) Comparaison des temps

Cette solution fait un peu plus confiance à l'utilisateur, et fait un large usage de la règle des 35%.

Supposons que le didacticiel possède un intervalle de confiance centré sur une valeur étalon représentative des vraies capacités du lecteur. **Notre thèse sera que si un lecteur lit trop rapidement ou trop lentement pour ces capacités, le temps chronométré s'éloignera sensiblement de l'étalon au point de sortir de l'intervalle de confiance. Ce test souple qui se base uniquement sur la comparaison entre des temps de lecture enregistrés sera à la base de tout les contrôles que nous utiliserons.** (Les calculs et décisions sous-jacentes sont détaillées plus loin dans les exercices d'évaluation du CF et du TMF.)

Le gros défaut de cette méthode est qu'on ne repère que les gros écarts par rapport à la moyenne des performances. Cette moyenne peut être faussée si, depuis le début, le lecteur lit mal ou sabote systématiquement. Ainsi, même la valeur étalon n'est par sûre à 100%.

CONCLUSION

De mon propre chef, je vais devoir fixer des valeurs pour les temps absurdes, et prendre des décision quant à la façon de diagnostiquer. De plus il est possible que les 35% de la fameuse règle, ne soient pas un bon intervalle. Il faut donc laisser la possibilité de modifier facilement ces valeurs et décisions.

Ainsi les 35%, les temps absurdes, et d'autres éléments-clés de diagnostic pourront être changés interactivement, et formeront l'ENVIRONNEMENT du didacticiel. (Pour plus de détails, voir la fin du chapitre 6).

Chapitre 5 : DECISIONS A LA BASE DU DIDACTICIEL

Dans ce chapitre, nous verrons successivement :

- Les méthodes de recherche des règles de diagnostic.
- La liste des hypothèses formant la base de mes décisions.
- Les normes de calcul du TMF.
- Les normes d'acceptation d'un temps de fixation.
- L'imprécision dans le calcul d'estimation des TMF

5.A) METHODES DE RECHERCHE DES REGLES DE DIAGNOSTIC

La première méthode que j'ai employée parce qu'elle me paraissait être la plus naturelle, ne tarda pas à se révéler mauvaise. Ayant une idée générale de ce que je devais diagnostiquer, j'ai choisi un peu arbitrairement tout type de résultats (premier, dernier, meilleur, moyenne, ...etc...) paraissant intéressants. Ensuite, j'essayais d'en tirer des conclusions en les associant. Mais le nombre de combinaisons est très élevé et certaines d'entr'elles sont absurdes quant à leur signification. De plus, cette méthode me forçait à créer de nouveaux types de résultats. Ce qui, par souci de cohérence, remettait en question les décisions précédentes. Le choix de cette méthode me fit perdre un bon mois. J'adoptais alors la seconde méthode.

La seconde méthode se révéla être beaucoup plus rapide, efficace et cohérente. J'ai commencé par définir précisément ce que je voulais diagnostiquer et puis seulement, j'ai choisi les types de résultats et leurs rapports significatifs permmettant ce diagnostic. Il restait alors à définir les intervalles de valeurs représentatifs de la qualité de la lecture. Quand la théorie de F.Richaudeau restait silencieuse sur un cas, j'ai laissé au lecteur la possibilité de tirer lui même les conclusions. Cela limite l'arbitraire de mes décisions.

5.B) LISTE DES HYPOTHESES DE BASE DU DIDACTICIEL

1 - Un lecteur adulte ne craignant pas les sanctions d'un tiers, préfère être conseillé plutôt que forcé.

2 - L'utilisateur est un lecteur moyen, ni prodige ni trop mauvais. Il ne commet ni sabotage systématique, ni d'erreurs continues depuis le début.

3 - L'attention nécessaire pour appuyer MACHINALEMENT sur une touche du clavier, est négligeable.

4 - Tout lecteur reconnaît un mot court et courant à sa forme.

5 - Pendant la lecture d'un texte, les temps de fixation appartiennent à une fourchette centrée sur une valeur moyenne. La largeur de la fourchette ne dépasse pas 35 % de cette moyenne. On accepte donc une certaine imprécision.

6 - Le T.M.F des lecteurs ayant un C.F large, est un peu plus long que pour ceux qui ont un C.F étroit.

7 - Le C.F varie avec la concentration et l'entraînement du lecteur. La largeur du CF n'est significative qu'à 3 signes près.

8 - L'entrainement d'un lecteur ne peut guère modifier son TMF, alors qu'il peut sensiblement faire évoluer le TMF. Le didacticiel cherchera donc à aggrandir le CF par l'entrainement.

9 - Le bon lecteur intègre le sens d'un bout de phrase reconnu dans un texte, pendant le mouvement des yeux.

10 - Les temps moyens nécessaires pour chacun des exercices suivant sont différents.

a) IDENTIFICATION DE MOTS SEMBLABLES.

En comparant avec un modèle, **IDENTIFIER** d'un coup d'oeil et au contour, un mot mélangé à d'autres qui lui ressemblent.

b) IDENTIFICATION PAR LECTURE EN COLONNE.

En un point de fixation, reconnaître et **COMPRENDRE** une phrase courte ou un mot PRIVE DE CONTEXTE.

c) LECTURE PAR POINT DE FIXATION DE TEXTES EN Y.

En lisant par point de fixation des textes **CONNUS**, reconnaître, comprendre et **INTEGRER** un bout de phrase.

d) LECTURE NATURELLE .

En lisant par points de fixation un texte **INCONNU**, comprendre et intégrer un bout de phrase.

EN (a), le temps nécessaire n'est pas un T.M.F, car il ne s'agit que de reconnaître un contour, de comparer un mot à un modèle, sans chercher à le comprendre.

En (b), le T.M.F comprend un temps de compréhension de mot ou de phrase courte, sans intégration.

En (c), Le temps moyen incorpore en plus la durée nécessaire pour intégrer un bout de phrase.

En (d), le temps moyen ne signifie rien, car le lecteur peut survoler des passages et freiner pour d'autres. Il dépend aussi de la difficulté du texte. Ce temps moyen ne peut pas être comparé à un T.M.F.

Seuls les temps moyens des exercices (b) et (c) , peuvent être considéres comme des T.M.F.

REMARQUE : Il existe une relation intéressante entre les exercices (b) et (c).

Si le T.M.F de (b) est plus petit que celui de (c), cela signifie que pour (c), le lecteur n'utilise pas le temps de mouvement des yeux pour intégrer le sens. Il n'ose pas aller de l'avant, il est pessimiste . Le plus important dans l'excercice (c), est de prendre l'habitude de lire des textes en Y par points de fixation. Le texte étant de gamme, on peut le lire plusieurs fois. Il ne serait donc pas pertinent d'évaluer la qualité de la lecture en posant des questions sur le contenu du texte. Ce n'est donc pas tout à fait une lecture naturelle, car on y attache moins d'importance à l'intégration du sens.

J'en déduis que, si le lecteur ose aller de l'avant dans un texte qu'il considère lui-même comme facile, le T.M.F de l'excercice (c) doit se rapprocher de celui de l'excercice (b) .

Par conséquent,

1) Si deux T.M.F résultant de la lecture par points de fixation de 2 textes CONNUS et jugés de difficulté égale par le lecteur, sont trop différents, le lecteur a sans doute été trop vite ou trop lentement lors de la lecture d'au moins un des deux textes.

2) Si la largeur d'un champ imposée pour un excercice de lecture de textes en Y et pour un excercice de lecture de mots en colonne, est égale au C.F du lecteur,

et SI le lecteur intègre le sens du bout de phrase pendant le mouvement des ses yeux,

ALORS, les T.M.F pour ces 2 exercices seront semblables .

Dans le cas contraire, le TMF de la lecture par points de fixation révèle que le lecteur n'intègre pas le sens des bouts de phrases pendant le mouvement des yeux.

5.C) NORME POUR LE CALCUL DU TMF

Pour contrôler certaines performances du lecteur, nous avons besoin d'une évaluation assez sûre de son TMF réel. Pour cela nous utilisons la règle des 35%.

5.C.1) Problème d'interprétation de la règle des 35%

F.Richaudeau écrit à la page 79:

‹‹ D'un test à l'autre, il y a des écarts maximaux de 35% dans la durée moyenne de fixation.››

Il faut définir 35% de quoi ?

5.C.2) Vocabulaire

- **MIN** : Temps de Fixation minimum POSSIBLE pour un lecteur donné.
- **MAX** : Temps de Fixation Maximum POSSIBLE pour un lecteur donné.
- **DIF** : Différence de temps entre MIN et MAX.
- **M** : Moyenne de MIN et MAX = (MIN + MAX) / 2
- **T** : Moyenne de tous les T.M.F calculés pour un lecteur donné.

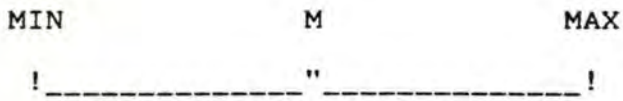
REMARQUE: A l'exception de T, ces temps et cette moyenne sont des représentations directes de la réalité que "Seul-le Bon-Dieu-connaît". Ce NE sont donc PAS des approximations évaluées sur base d'expériences.

5.C.3) Mon interprétation

J' interpréterai la phrase de F.Richaudeau comme ceci:

" L'écart entre MIN ET MAX ne dépasse jamais 35% de leur moyenne M."

Schéma 0



avec,
MAX = (117,5% * M),
MIN = (82,5% * M), et
DIF ‹= (35% * M).

Donc, pendant la lecture d'un texte, les variations entre les temps de fixation ne dépasse pas 35%. Mais, selon le temps séparant les mesures,l'entraînement, la forme du lecteur, les variations du TMF peuvent ne pas dépasser 10%, ou au contraire sauterla barre des 35%. Malgré cette imprécision, le didacticiel a besoin d'une règle pour décider; donc je décide que toute mesure située dans l'intervalle des 35% révèle que le lecteur travaille correctement.

5.C.4) Problème d'estimation de l'intervalle

Ma première approche de ce problème m'a conduit à une méthode trop peu résistante aux erreurs ayant lieu lors des premières estimations. C'est pourquoi, mon promoteur Mr. Cherton m'a proposé cette nouvelle approche, moins dépendante des premières valeurs retenues.

Il faut essayer de trouver une valeur qui ressemble à M. L'approche se fera en deux grandes étapes.

1ère étape :

- Mémoriser les 100 Premiers temps de fixation non-absurdes

REMARQUES :
a) Pour des raisons qui sont expliquées plus loin dans le paragraphe concernant les normes d'acceptation des temps de fixation, on insérera au début de cette liste, un temps de fixation égal à 250 millisecondes.
b) Un temps absurde trop court pour un point de fixation est de l'ordre de 100 millisecondes;
c) Un temps absurde trop long pour un point de fixation est de l'ordre de 400 millisecondes;
d) Un temps absurde trop long pour 2 points de fixation est de l'ordre de 800 millisecondes + 20 millisecondes si on compte le temps de mouvement des yeux;

- Effectuer la moyenne MOY de ces temps.

- Effacer les temps plus grands que (117,5% * MOY),

et les temps plus petits que (82,5% * MOY).

- Effectuer la moyenne T des temps restant.

A la fin de cette étape, on possède une valeur de T considérée comme représentative du TMF réel du lecteur.

ERRATUM

Malgre toute notre attention, une faute de frappe importante s'est glissée dans l'équation d'ACCEPTATION. Veuillez nous en excuser !...

P.49:

```
Soit A = ABSOLUTE [ MAXI(MX.Ti)  - ( (i-1)T+Ti )/i ]
Soit B = ABSOLUTE [ MINI(MN.Ti)  - ( (i-1)T+Ti )/i ]
     Si A <=   17.5 * [ (i-1)T+Ti) / (i*100) ]  ET
     Si B <=   17.5 * [ (i-1)T+Ti) / (i*100) ]
  Alors  MX = MAXI(MX.Ti)
         MN = MINI(MN.Ti)
          T = [(i-1)T + Ti] / i
```

2nde étape :

A partir d'ici , on oublie la liste des temps.

Les informations à retenir seront :

- La moyenne **T**.
- Le plus long temps accepté jusqu'ici: **MX**
- Le plus court temps accepté jusqu'ici: **MN**

Alors pour affiner la valeur de T, a chaque nouveau temps Ti calculé, on procèdera ainsi:

Soit MAXI et MINI, 2 fonctions rendant respectivement le maximum, (resp.minimum) de leurs 2 paramètres.

Soit A = ABSOLUTE [MAXI(MX,Ti) - ((i-1)T-Ti)/i]

Soit B = ABSOLUTE [MINI(MN,Ti) - ((i-1)T-Ti)/i]

Si A <= 17,5 * [(i-1)T+Ti) / (i*100)] ET

Si B <= 17,5 * [(i-1)T+Ti) / (i*100)]

Alors MX = MAXI(MX,Ti)

MN = MINI(MN,Ti)

T = [(i-1)T + Ti] / (i*100)

On appellera cette équation : " **L'EQUATION D'ACCEPTATION** ". Elle a l'avantage d'équilibrer l'importance donnée aux nouvelles valeurs et aux anciennes. L'évaluation du TMF, peut donc évoluer sans à-coups brusques, et sans être trop liées aux valeurs de départ; Malheureusement, elle ne repère que les gros écarts par rapport à la moyenne des performances du lecteur.

Ce problème est inévitable. Il est par conséquent très important que le lecteur fasse correctement les exercices. Sa participation et sa bonne volonté sont indispensables pour obtention une valeur de référence fiable dont nous connaissons l'importance dans le type de contrôle que nous effectuons.

5.D) NORME POUR L'ACCEPTATION D'UN TEMPS DE FIXATION

Pour décider si un lecteur a reconnu un mot à sa forme, sans le forcer à recomposer un mot, il faut une norme permettant d'accepter ou refuser ses temps moyens de lecture de mots. La solution proposée répond à cette attente, mais se révèlera mauvaise, si dés le début le lecteur sabote ou fait continuellement les même erreurs.

1)Observations

Si nous possédons une approximation assez sûre du TMF, l'équation d'acceptation nous permet, à la fois, d'affiner cette approximation et de juger si le temps de lecture d'un mot est représentatif d'un seul PF ou non.

Remarquons que sans cette approximation assez sûre du TMF, il est difficile de poser le même jugement.

En effet, nous venons de voir qu'au début (avant de posséder les 100 premières mesures) pour calculer le TMF, on retient tous les temps non absurdes.(Ceux compris entre 100 et 400 millisecondes.). Cet écart de 300 millisecondes est trop grossier pour se rendre compte si le lecteur a lu un mot en 2 PF. En effet, si le TMF réel du lecteur est de 200 millisecondes, et qu'il lit un mot en 2 PF, le temps calculé sera a peu près de 400 millisecondes. Etant dans les normes de calcul du TMF, il serait malgré tout accepté comme un temps de fixation. Nous devons donc fixer une norme propre à l'acceptation des temps.

2)Normes

SI la liste contient déjà 100 éléments, on utilise sytématiquement l'équation d'acceptation en utilisant l'évaluation actuelle du TMF.

Si la liste contient moins de 100 éléments, on considérera , malgré que ce ne soit pas une estimation sûre, que le TMF actuel est la moyenne de tous les éléments de la liste. Par conséquent, on jugera qu'un mot a été lu en un PF, en utilisant aussi l'équation d'acceptation.

Il reste le problème de la première valeur retenue, qui risque de fausser les estimations futures. En effet, on acceptera la toute première valeur calculée, si elle n'est pas absurde. Or le lecteur avait peut-être lu trop vite ou trop lentement. Il est trop dangereux de considérer ce premier temps comme un TMF réel. Il faudrait pondérer ce risque en estimant que le TMF doit être la moyenne de ce temps et d'une valeur de référence. Comme F.Richaudeau a calculé qu'en moyenne le temps de fixation de chacun tourne autour du quart de seconde, je fixerai la valeur de référence à 250 millisecondes. Cette valeur sera toujours la première de la liste des 100.

REMARQUES :

1 - Les 2 normes se ressemblent très fort sauf pour le traitement de la première valeur retenue. Le lecteur peut se demander pourquoi je veux définir 2 normes alors qu'elles sont presqu'identiques. Je répondrai que l'important n'est pas l'algorithme des 2 normes mais leurs concepts sous-jacents. Il faut faire la différence entre le besoin d'affinement du TMF, et le besoin de réagir interactivement à une performance du lecteur.

En effet, au début, le didacticiel n'évalue pas bien le lecteur. Par conséquent, tous les temps non-absurdes peuvent être considérés comme représentatifs des performances du lecteur; Mais, même si ce dernier n'a fait que 2 exercices, le didacticiel doit réagir si la nouvelle performance diffère fort des 2 premières. Remarquons que, même si on change les algorithmes à la base de ces normes, les concepts ne changent pas.

2 - Il y a une certaine imprécision dans le calcul d'estimation des TMF.

Les imprécisions sont dues à la méthode et à l'outil utilisés.

a) - Le chrono est d'une imprécision allant de 0 à 20 millisecondes selon les cas.

b) - Le temps moyen évalué par le didacticiel pour la lecture de mots en colonne, n'est pas un T.M.F pur. La valeur obtenue est le T.M.F + le temps moyen de mouvement des yeux.

Ce n'est pas très grave car le temps de mouvement des yeux est petit par rapport au TMF. Nous pouvons donc le négliger.

CONCLUSIONS

Le point le plus important à retenir est que le besoin d'un contrôle souple des performances, nous conduit à décider de la validité d'un exercice sur base d'un comparaison des temps.

Une performance ne sera jugée valable que si le temps correspondant est proche d'un temps de référence.

Chapitre 6 : DESCRIPTION DES EXERCICES IMPLEMENTES

Dans ce chapitre, nous verrons successivement :

- Présentation générale du didacticiel.
- Lecture en colonne.
- Lecture par points de fixation.
- Evaluation du TMF et du CF.
- Reconnaissance de mots au contour.
- Eviter les confusions de mots.
- Evaluation des tendances à lire mot à mot.
- Evaluation des tendances a subvocaliser.
- Lecture naturelle.
- Des valeurs à changer.

6.A) PRESENTATION GENERALE DU DIDACTICIEL

6.A.1) Instruments

Le didacticiel va calculer tous les temps automatiquement. Il se charge aussi de modifier fréquemment le contenu des exercices. Lors de l'exercice traitant de la confusion des mots, le lecteur signalera qu'il a régressé en tapant la touche "ENTER". Enfin le cache, utilisé pour la lecture en colonne, sera simulé par l'apparition successive des mots à l'écran.

Le lecteur n'a donc plus besoin de chronomètre,ni de crayon, de cache ou de longues listes d'exercices.

6.A.2) Informations

- Quand le chrono est déclenché le lecteur ne doit pas perdre son temps à chercher comment fonctionne l'exercice. C'est pourquoi, avant, il peut accéder à un texte lui expliquant comment l'effectuer.

- Puisqu'il est nécessaire que le lecteur soit conscient de l'utilité d'un exercice, il peut accéder à un texte lui expliquant le théorie défendue par F.Richaudeau, et à d'autres décrivant les effets de chaque exercice.

6.A.3) Libertés

1) Lors de la première session , afin que le didacticiel possède un minimum d'informations sur le lecteur, ce dernier sera forcé de subir certaines évaluations, avant de pouvoir choisir les exercices qu'il préfère.

Ce sont : -l'évaluation des tendances à lire mot à mot
-l'évaluation du TMF et CF.
- l'évaluation des capacités à reconnaître un mot à son contour.
-l'évaluation du nombre de signes lus à l'heure pour une lecture naturelle.

De plus si on n'a pas encore diagnostiqué une perte des tendances à lire mot à mot, il y aura reévaluation forcée trois sessions après la dernière évaluation.

2) Si, en début de session, l'analyse des résultats de lecture en colonne ou par points de fixation, montrent une modification du CF, la reévaluation sera forcée.

En dehors de ces deux cas le lecteur fait ce qu'il veut.

6.A.4) Assiduité

- Il est important que le lecteur soit régulier, on retiendra donc toujours la date de dernière session. Si les performances sont en baisse pour un exercice, et que l'utilisateur n'y avait plus travaillé depuis au moins 7 jours, on le lui fera remarquer. On insiste sur le fait que plus les résultats d'un exercice sont anciens, moins ils conservent de significations réelles. Cela risque de fausser les résultats du didacticiel.

- Si le lecteur veut sortir après seulement 10 minutes, on lui fait remarquer son manque de persévérance.

- Si le lecteur travaille depuis une heure, on lui signale qu'il ne faut pas forcer. Je ne fixe pas un temps optimal de durée de session car il n'y a pas moyen de le connaître pour chaque lecteur.

- S'il sort sans avoir fait au moins une fois l'exercice de lecture en colonne, de lecture par points de fixation, on le lui fait remarquer. On fera de même avec l'exercice de reconnaissance de mot au contour, s'il n'est pas jugé que le lecteur à une habileté de perception suffisante.

A la session suivante on lui rappellera ses oublis.

6.A.5) Types d'exercices

On en distingue deux :

1- Ceux qui ne servent qu'à évaluer les performances du lecteur. On y retrouve l'exercice d'évaluation du CF et du TMF, ainsi que celui d'évaluation des tendances à lire mot à mot. Ils se caractérisent par leur aspect contraignant.

2- Ceux qui servent d'abord à entrainer le lecteur. Ils permettent quand même d'évaluer les progrès accomplis.

6.A.6Conseils

Il y a deux types de diagnostic :

- Celui qui porte sur le résultat d'un exercice qui vient d'être effectué.

- Celui qui analyse l'ensemble des résultats mémorisés d'un exercice.

6.A.7) Capacité maximale

Pour des raison de présentation à l'écran, j'ai fixé à 26 caractères, la largeur maximale des demi-phrases des textes en Y.

AVERTISSEMENTS:

- Il est nécessaire d'avoir lu le chapitre 2.B, pour bien comprendre la description suivante des exercices.

- J'insiste sur la notion de diagnostic. Nous en avons défini deux types :

a)**Le diagnostic qui fait suite à un exercice.**

Il se contente de commenter la performance courante en la comparant à l'ensemble des résultats mémorisés pour cet exercice.

b)**Le diagnostic général.**

Il porte sur l'ensemble des résultats enregistrés pour chaque exercice. Il permet d'évaluer le niveau actuel moyen du lecteur et ses progrès.

Remarquons que les diagnostics sont à la fois des commentaires du didacticiel et l'affichage de résultats. Ainsi le lecteur a toujours la possibilité de négliger les avis du didacticiel et de porter lui même un jugement sur ses propres résultats.

6.B) LECTURE EN COLONNE

6.B.1) Buts

- Améliorer la capacité à reconnaître les mots à la forme.
- Elargir le champ de fixation.
- Repérer cet élargissement par observation du temps de lecture de mots plus longs que le dernier CF retenu.

6.B.2) Instruments et présentation

- D'abord apparaissent 2 colonnes vides; le milieu de chaque colonne est désigné par un point représentant l'endroit où le lecteur devra fixer les yeux.

- Le lecteur tape sur la barre d'espacement pour faire apparaître le premier mot ou pour signaler qu'il a reconnu le mot courant.

- Le cache est simulé par ces apparitions successives de mots. Le lecteur ne lit que le dernier affiché.

- Le chrono se déclenche à l'apparition du premier mot, et s'arrête après la lecture du dernier.

6.B.3) Déroulement

- le lecteur lit aussi vite que possible une liste de mots. La première moitié de cette liste comporte des mots d'une largeur plus petite ou égale au CF actuel. La largeur des mots de la seconde moitié, sera comprise entre le CF et le CF plus un pas de 3 ou 4 caractères.

- Si la première moitié de la liste a été lue incorrectement, l'exercice s'arrête aussitôt. En effet,le lecteur n'ayant pas travaillé correctement avec les mots faciles, il est inutile de persévérer car les mots proposés par la suite sont plus compliqués.

6.B.4) Valeurs retenues

- Liste des 10 dernièrs temps, (et leur numéro de session) retenus pour la lecture des mots courts. **(ABRV : LC)**

- Liste des 10 dernièrs temps, (et leur numéro de session) retenus pour la lecture des mots longs. **(ABRV : LL)**

REMARQUES :
- A l'ajout du 11 ième élément en début de liste, le premier est éliminé.
- Il faut que la liste contienne au moins 3 éléments pour que sa moyenne soit significative.
- L'avantage à ne retenir que les 10 dernières valeurs, est qu'on conserve une vue récente des capacités du lecteur

- Nombre de temps contenus dans les 2 listes.

- La meilleure moyenne de la LL.

TOUTES CES VALEURS SONT REMISES à 0, APRES CHAQUE EVALUATION DU TMF ET DU CF.

6.B.5) Intervalles significatifs des MOYENNES des temps de lecture

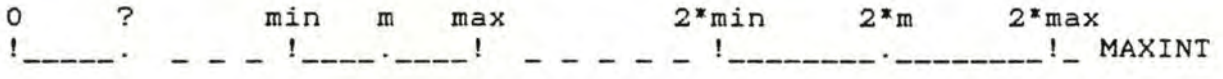
Les temps dont nous parlons, sont les temps moyens de lecture de mots ou de phrases dont la largeur ne dépasse pas 26 caractères.

Soit m, l'évaluation actuelle du TMF.

Soient **min** = (82,5% * m) et

max = (117,5% * m)

Schéma 1.



0‹--›? : Temps humainement impossible.

? ›--‹ min : Temps trop petit pour un lecteur donné.

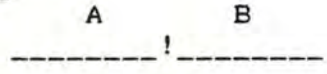
min ‹--› max : TMF pour un lecteur donné. Cet intervalle est appelé **"INTERVALLE DE CONFIANCE"**.

2*min ‹--‹ MAXINT : Temps moyen pour au moins 2 points de fixation.

MAXINT : Temps absurdément trop élevé.

max ›--‹ 2*min : Intervalle trop long pour un point de fixation; trop court pour 2. On peut diviser cet intervalle, en 2 moitiés A et B.

Schéma 2.



On posera qu'en A, on a lu plus souvent en 1 seul point de fixation qu'en 2. Et l'inverse pour B.

Malgré un souci de souplesse, ce schéma laisse apparaître un certain arbitraire, une décision étant préférée à une autre, pour une différence de quelques millisecondes. Cela est incontournable.

6.B.6) Idées principales de diagnostic

- Si, en moyenne, le temps de lecture des mots plus larges que le dernier CF évalué, entre dans l'intervalle de confiance du TMF étalon, le CF a sûrement grandit.

- Si, en moyenne, le temps de lecture des mots d'une largeur au plus égale au dernier CF évalué sort de l'intervalle de confiance, on soupçonne que le CF **OU LE TMF** a changé.

6.B.7) Diagnostics et réactions après l'exercice

Soit Tc, (resp: Tl) le temps moyen de lecture des mots courts (resp : longs) pour l'exercice.

On affiche toujours le TMF, le CF, Tc, Tl, les moyennes de LC et LL, la meilleure valeur de LL. On signale toujours si ces moyennes sont significatives. Si le lecteur n'a pas travaillé depuis 7 jours, on l'indique.

1- Si Tc ou Tl est absurde, on le fait remarquer. Rien n'est mémorisé. Si Tc n'est pas absurde , il est utilisé pour affiner l'évaluation du TMF.

2- Si Tc est non absurde mais en dehors de l'intervalle de confiance, le lecteur n'a pas lu les mots courts en un point de fixation et le didacticiel a arrêté l'exercice avant de passer à la lecture des mots longs plus difficiles à lire ,alors:

Tc est inséré dans LC.
Tl prend la valeur 2*Tc, et est inséré dans LL.

Si le lecteur n'avait plus fait d'exercices depuis 7 jours, il faut signaler qu'il lirait mieux s' il s'exerçait plus souvent.

Le lecteur doit se forcer à lire plus vite, ou moins vite, selon la position de Tc dans le schéma 1.
Si la moyenne de Lc est significative, et dépasse l'intervalle de confiance du TMF, il faut INSISTER TRES FORT pour que le lecteur reévalue son CF. Il a pris l'habitude de lire trop lentement par rapport à ce qu'on connait de ses capacités.
Si la moyenne de Lc est significative, et passe sous l'intervalle de confiance du TMF, IL FAUT INSISTER TRES FORT pour que le lecteur reévalue son CF. Il a pris l'habitude de lire trop vite pour ce qu'on connait de ses capacités.

3- Si Tc est dans l'intervalle de confiance,
alors:
Tc est inséré dans LC.
Tl est inséré dans LL.

Le lecteur a lu correctement les mots courts.

On compare Tl à la moyenne de LL. On signale au lecteur qu'il a lu plus ou moins vite par rapport à sa moyenne.

a) Si la nouvelle valeur de la moyenne de LL avec Tl, devient la plus petite, on le signale au lecteur.

b) Si Tl est dans l'intervalle ' 2*max ›--› MAXINT, signaler au lecteur qu'il lit les mots en plus de 2 points de fixation. Il y a un gros effort à faire.

c) Si Tl est dans l'intervalle ' 2*min ‹--› 2*max', il faut signaler au lecteur qu'il doit se forcer à lire les mots longs en un seul point de fixation.

d) Si Tl est dans l'intervalle décrit au schéma 2.(cfr infra.)
alors si Tl est dans B , signaler au lecteur qu'il lit les mots plus souvent en 2 points de fixation qu'en 1;
mais si Tl est dans A, signaler au lecteur qu'il lit plus souvent en 1 point de fixation qu'en 2.

e) Si Tl est dans l'intervalle de confiance, signaler au lecteur qu'il a lu en seul point de fixation les mots longs. Alors si la nouvelle valeur de LL passe dans cet intervalle grâce à Tl, il faut INSISTER TRES FORT pour que le lecteur reévalue son CF.
ATTENTION : Si Tl est plus petit que Tc, faire remarquer qu'il a sans doute lu les mots longs un peu trop vite.

6.B.8)Diagnostic général

Le CF et le TMF, les moyennes de LC et LL, et la meilleur moyenne de LL, sont affichées. On signale toujours si ces moyennes sont significatives.S'il n'a pas travaillé depuis 7 jours, on le fait remarquer au lecteur.

1- Si la moyenne de LC est en dehors de l'intervalle de confiance,alors
si on vient de commencer la session et que la moyenne est significative, LA REEVALUATION DU CF SERA FORCEE ;
sinon il faut INSISTER TRES FORT sur le fait que le lecteur doit reévaluer son CF et qu'il doit se forcer à lire plus vite, ou moins vite, selon la position de la moyenne dans le schéma 1.
Si le lecteur n'a plus fait d'exercices depuis 7 jours, il faut signaler qu'il lirait mieux s' il travaillait plus régulièrement.

2- Si la moyenne de LC est dans l'intervalle de confiance alors:
a) Si la moyenne de LL est plus petite que celle de LC, il faut prévenir l'utilisateur qu'il lit sans doute trop vite les mots longs.

b) Si la moyenne de LL se situe dans l'intervalle 2*max ›--› MAXINT, signaler au lecteur qu'il lit les mots en plus de 2 points de fixation. Il y a un gros effort à faire.

c) Si la moyenne de LL se situe dans l'intervalle : 2*min ‹--› 2*max, il faut signaler au lecteur qu'il doit se forcer à lire les mots longs en un seul point de fixation.

d) Si la moyenne de LL est dans l'intervalle décrit au shéma 2.(cfr infra.)
alors si cette moyenne est dans B , signaler au lecteur qu'il lit les mots plus souvent en 2 points de fixation qu'en 1 seul; mais si cette moyenne est dans A, signaler au lecteur qu'il lit plus souvent en 1 point de fixation qu'en 2.

3- Si la moyenne de LL est est dans l'intervalle de confiance, alors
si on vient de commencer la session et que la moyenne est significative, on FORCE LA REEVALUATION DU CF ;
sinon il faut INSISTER TRES FORT auprès du lecteur pour qu'il reévalue son CF.

6.C) LECTURE PAR POINTS DE FIXATIONS

6.C.1) Buts

- Donner l'habitude de lire par points de fixation, en intégrant le sens pendant le mouvement des yeux.
- Elargir le champ de fixation.
- Repérer cet élargissement par observation du temps de lecture de mots plus longs que le dernier CF retenu.

6.C.2) Instruments et présentation

- D'abord apparait une ligne au sommet de l'écran.Le lecteur ne devra s'intéresser qu'à la phrase située juste en dessous.
- Le lecteur tape la barre d'espacement pour faire apparaître le texte enforme d'Y.
- Pour signaler qu'il a lu la phrase courante, le lecteur tape BE. Cela fait remonter le texte d'un cran. La ligne courante est donc remplacée par sa suivante.
- Le chrono se déclenche à l'apparition du texte et s'éteint après la lecture de la dernière phrase.

6.C.3) Déroulement

- le lecteur lit aussi vite que possible le texte en conservant l'habitude de lire en 2 points de fixation.
- A peu près la première moitié du texte comporte des demi-phrases d'une largeur plus petite ou égale au CF actuel. La largeur des demi-phrases de la seconde moitié, sera comprise entre le CF et le CF plus un pas de 3 ou 5 caractères.
- Si la première moitié du texte a été lue incorrectement, l'exercice s'arrête aussitôt. En effet,le lecteur n'ayant pas travaillé correctement avec les demi-phrases faciles, il est inutile de persévérer car les demi-phrases proposées par la suite sont plus compliquées.

6.C.4) Valeurs retenues

- Liste des 10 dernièrs temps, (et leur numéro de session) retenus pour la lecture des mots courts. **(ABRV : LC)**

- Liste des 10 derniers temps, (et leur numéro de session) retenus pour la lecture des mots longs. **(ABRV : LL)**

REMARQUES :
- A l'ajout du 11 ième élément en début de liste, le premier est éliminé.
- Il faut que la liste contienne au moins 3 éléments pour que sa moyenne soit significative.
- L'avantage à ne retenir que les 10 dernières valeurs, est qu'on conserve une vue récente des capacités du lecteur

- Nombre d'éléments dans les 2 listes.

- La meilleure moyenne de la LL.

TOUTES CES VALEURS SONT REMISES à 0, APRES CHAQUE EVALUATION DU TMF ET DU CF.

6.C.5)Intervalles significatifs des MOYENNES des temps de lecture

Les temps dont nous parlons, sont les temps moyens de lecture de demi-phrases dont la largeur ne dépasse pas 26 caractères.

Soit m, l'évaluation actuelle du TMF.

Soient **min** = (82,5% * m) et

max = (117,5% * m)

max+ = (122,5% * m) { J'ai fixé cette valeur arbitrairement}

Schéma 3.

0‹--›? : Temps humainement impossible.

? ›--‹ min : Temps trop petit pour un lecteur donné.

min ‹--› max : Temps pour lire en un point de fixation pour un lecteur donné. Cet Intervalle est appelé **"intervalle de confiance."**

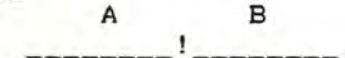
max ›--› max+ : **Temps pour lire en un point de fixation MAIS SANS INTEGRER LE SENS DES PHRASES PENDANT LE MOUVEMENT DES YEUX.**

2*min ‹--‹ MAXINT : Temps moyen pour au moins 2 points de fixation.

MAXINT : Temps absurdément trop élevé.

max+ ›--‹ 2*min : Intervalle trop long pour un point de fixation, avec intégration du sens durant le mouvement des yeux;trop court pour 2. On peut diviser cet intervalle, en 2 moitiés A et B.

Schéma 4.



On posera qu'en A, on a lu plus souvent en 1 seul point de fixation qu'en 2. Et l'inverse pour B.

Malgré un souci de souplesse, ce schéma laisse apparaître un certain arbitraire, une décision étant préférée à une autre, pour une différence de quelques millisecondes. Cela est incontournable.

6.C.6) Idées principales de diagnostic

- Si, en moyenne, le temps de lecture des mots plus larges que le dernier CF évalué entre dans l'intervalle de confiance du TMF étalon, le CF a sûrement grandit.

- Si, en moyenne, le temps de lecture des demi-phrases d'une largeur au plus égale au dernier CF évalué sort de l'intervalle de confiance allant de MIN à max+, on soupçonne que le CF **OU LE TMF** a changé.

- Si, en moyenne, le temps de lecture des demi-phrases d'une largeur au plus égale au dernier CF évalué, passe de l'intervalle allant de min à max+ a celui qui s'étend de min à max, on soupçonne que le lecteur intègre le sens des phrases pendant le mouvement des yeux.

6.C.7) Diagnostics et réactions après l'exercice

Soit **Tc**, (resp: **Tl**) le temps moyen de lecture des mots courts (resp : longs) pour l'exercice.

On affiche toujours le TMF, le CF, Tc, Tl, les moyennes de LC et LL, la meilleure valeur de LL. On signale toujours si ces moyennes sont significatives. S'il n'a pas travaillé depuis 7 jours on le fait remarquer.

1- Si Tc ou Tl est absurde, on le fait remarquer. Rien n'est mémorisé.

2- Si Tc est non absurde mais en dehors de l'intervalle allant de min à max+, le lecteur n'a pas lu les demi-phrases courtes en un point de fixation et le didacticiel a arrêté l'exercice avant de passer à la lecture des demi-phrases longues plus difficiles à lire ,alors:

Tc est inséré dans LC.
Tl prend la valeur 2*Tc, et est inséré dans LL.

Si le lecteur n'avait plus fait d'exercices depuis 7 jours, il faut signaler qu'il lirait mieux s' il s'exerçait plus souvent.

Le lecteur doit se forcer à lire plus vite, ou moins vite, selon la position de Tc dans le schéma 1.

Si la moyenne de Lc est significative, et dépasse l'intervalle allant de min à max+, il faut INSISTER TRES FORT pour que le lecteur reévalue son CF. Il a pris l'habitude de lire trop lentement par rapport à ce qu'on connait de ses capacités.

Si la moyenne de Lc est significative, et passe sous l'intervalle de confiance du TMF, il faut INSISTER TRES FORT pour que le lecteur reévalue son CF. Il a pris l'habitude de lire trop vite par rapport à ce qu'on connait de ses capacités.

3- Si Tc est dans l'intervalle allant de min à max+, alors:

Tc est inséré dans LC.
Tl est inséré dans LL.

Signaler qu'il a lu correctement les mots courts.

Mais si TC est se situe entre max et max+, il faut insister pour que le lecteur intègre pendant le mouvement des yeux.

On compare Tl à la moyenne de LL. On signale au lecteur qu'il a lu plus ou moins vite par rapport à sa moyenne.

Si la nouvelle valeur de la moyenne de LL avec Tl, devient la plus petite, on le signale au lecteur.

a) Si Tl est dans l'intervalle ' 2*max ›--› MAXINT, signaler au lecteur qu'il lit les mots en plus de 2 points de fixation. Il y a un gros effort à faire.

b) Si Tl est dans l'intervalle ' 2*min ‹--› 2*max', il faut signaler au lecteur qu'il doit se forcer à lire les demi-phrases longues en un seul point de fixation.

c) Si Tl est dans l'intervalle décrit au schéma 4.(cfr infra.)

alors si Tl est dans B , il faut signaler au lecteur qu'il lit les demi- phrases plus souvent en 2 points de fixation qu'en 1;

mais si Tl est dans A, signaler au lecteur qu'il lit plus souvent en 1 point de fixation qu'en 2.

d) Si Tl est dans l'intervalle min ‹--› max+, signaler au lecteur qu'il a lu en seul point de fixation les demi-phrases longues. Alors si la nouvelle valeur de LL, passe dans cet intervalle grâce à TL, INSISTER TRES FORT pour qu'il reévalue son CF.

ATTENTION : Si Tl est plus petit que Tc, il faut faire remarquer au lecteur qu'il a sans doute lu les demi-phrases longues un peu trop vite.

6.C.8) Diagnostic général

Le CF et le TMF, les moyennes de LC et LL, et la meilleur moyenne de LL, sont affichées. On signale toujours si ces moyennes sont significatives. S'il n'a pas travaillé depuis 7 jours on le fait remarquer.

1- Si la moyenne de LC est en dehors de l'intervalle allant de min jusque max+, alors si on vient de commencer la session et que la moyenne est significative, LA REEVALUATION DU CF EST FORCEE;

sinon il faut INSISTER TRES FORT sur le fait que le lecteur doit se forcer à lire plus vite, ou moins vite, selon la position de la moyenne dans le schéma 1.

Si le lecteur n'a plus fait d'exercices depuis 7 jours, il faut signaler qu'il lirait mieux s'il travaillait plus régulièrement.

2- Si la moyenne de LC est dans l'intervalle de confiance, alors:

a) Si la moyenne de LL est plus petite que celle de LC, il faut prévenir l'utilisateur qu'il lit sans doute trop vite les mots longs.

b) Si la moyenne de LL se situe dans l'intervalle : 2*max ›--› MAXINT, signaler au lecteur qu'il lit les mots en plus de 2 points de fixation. Il y a un gros effort à faire.

c) Si la moyenne de LL se situe dans l'intervalle : 2*min ‹--› 2*max, il faut signaler au lecteur qu'il doit se forcer à lire les mots longs en un seul point de fixation.

d) Si la moyenne de LL est dans l'intervalle décrit au shéma 4.(cfr infra.)

alors si cette moyenne est dans B , signaler au lecteur qu'il lit les demi-phrases plus souvent en 2 points de fixation qu'en 1 seul;

mais si cette moyenne est dans A, signaler au lecteur qu'il lit plus souvent en 1 point de fixation qu'en 2.

e) Si la moyenne de LL est est dans l'intervalle de confiance alors,

si on vient de commencer la session et que la moyenne est significative, la REEVALUATION DU CF EST FORCEE;

sinon il faut INSISTER TRES FORT auprès du lecteur pour qu'il reévalue son CF.

REMARQUE SUR LES DIAGNOSTICS :

Il peut arriver que le lecteur demande un diagnostic sur l'ensemble des résultats, et qu'on obtienne des résultats contradictoires pour la lecture en colonne et pour celle par points de fixation. Ce diagnostic sera obtenu par l'analyse comparative des LL et LC des deux exercices.

Il faudra expliquer les raisons de cette contradiction afin que le lecteur ne se demande pas pourquoi le didacticiel lui dit à la fois qu'il lit trop vite et trop lentement.

Pour expliquer la contradiction, c'est la cause la plus importante selon l'ordre donné ci-dessous qui sera avancée.

Voici les arguments par ordre décroissant d'importance:

1 - Les moyennes de LL et LC pour au moins un des exercices ne sont pas significatives.

2 - Les éléments des deux LC ont des degrés d'ancienneté différents, ou en d'autres mots, ils n'ont pas le même âge. Il va de soi que la liste la plus récente rend mieux compte du niveau actuel du lecteur.

3 - Une des listes est plus longue que l'autre, donc le lecteur a préféré faire un des exercices plus souvent que l'autre.

Si aucun de ces 3 cas n'est rencontré, le didacticiel préviendra le lecteur qu'il ne sait pas analyser les raisons qui conduisent à ces contradictions.

Pour avancer le deuxième argument, on a besoin de connaître l'âge relatif des listes LC des deux exercices de gamme. Pour cela nous allons définir une méthode de calcul.

Soit LC1 (resp: LC2) la LC de lecture en colonne (resp :de lecture par points de fixation).

Soient Cpt1 (resp: Cpt2) un compteur associé à LC1 (resp: LC2).

A chaque élément de LC1 et LC2 est associé un numéro de session. Nous comparerons chaque n° de session de LC1 avec son correspondant de LC2. Chaque fois que le numéro de session d'un élément de LC1 sera plus grand que son correspondant dans LC2, on incrémentera de 1 le compteur Cpt1. Dans le cas contraire, c'est le compteur Cpt2 qui sera augmenté.

Si Cpt1 est plus grand que Cpt2, à la fin de la comparaison, on soupçonnera que les éléments de LC1 sont plus récents que ceux de LC2. Et vice-versa.

EX)

	LC2		LC1	Réactions
n° de session	1	<	2	cpt1 + 1
	1	<	2	cpt1 + 1
	3	=	3	rien
	4	>	3	cpt2 + 1
	4	>	3	cpt2 + 1
	4	>	3	cpt2 + 1

Résultat: cpt1 = 2 et cpt2 = 3, DONC LC2 est une liste plus jeune.

Remarquons que si cette méthode est très simple, elle n'est pas fort fiable. Rien n'assure quelle découvre à coup sûr la liste la plus jeune.

Cependant, il est peut probable que nous devrons nous en servir. En effet, elle n'est utile que si les résultats de la lecture en colonne sont en contradiction avec ceux de la lecture par points de fixation. Or cette contradiction n'apparaîtra que si le lecteur fait un exercice plus souvent que l'autre. Mais comme **à chaque session**, le didacticiel insiste pour que le lecteur effectue les deux exercices, ce serait de la mauvaise volonté de la part du lecteur que de ne pas suivre ce conseil. Il n'y aura donc de risque que si le lecteur ne travaille pas sérieusement.

6.D) EVALUATION DU TMF ET DU CF

6.D.1) Buts

- Affiner la valeur du TMF.
- Evaluer le CF courant du lecteur.

6.D.2) Instruments et présentation

- D'abord apparaissent 2 colonnes vides; le milieu de chaque colonne est désignée par un point représentant l'endroit où le lecteur devra fixer les yeux.
- Le lecteur tape la barre d'espacement pour faire apparaître le premier mot, ou pour signaler qu'il a reconnu le mot courant.
- Le cache est simulé par cette apparition successive de mot. Le lecteur ne lit que le dernier affiché.
- Le chrono se déclenche à l'apparition du premier mot, et s'arrête après la lecture du dernier.
- Les phrases courtes ou mots utilisés ne dépassent pas 26 caractères.

6.D.3) Déroulement

- le lecteur lit aussi vite que possible une liste de 50 mots, ne dépassant pas 5 caractères.
- Il lira ensuite une liste de mots larges de 6 à 8 caractères.
- A chaque itération, on incrémentera la largeur des mots de 1 à 3 caractères.
- Si le TMF pour une liste de mot est absurde ou indique grâce à la "norme d'acceptation d'un temps de fixation" que l'utilisateur a lu en plus d'un point de fixation , on lui reproposera une liste avec des mots de même longueur. S'il réussit, il passe à la liste suivante; sinon le test se termine.

6.D.4) Valeurs retenues

- CF courant
- Le plus petit CF qu'a eut le lecteur.
- Le plus grand CF qu'a eut le lecteur.
- Les 100 premiers TMF enregistrés.
- La moyenne de tous les TMF enregistrés.

dans la norme de calcul DU TMF.

6.D.5) Diagnostics simples et réactions

Le CF est considéré égal à la largeur maximum des mots de la dernière liste lue correctement.

Le TMF est calculé selon la "norme pour le calcul du TMF" donnée plus haut.

Si le nouveau CF est différent de l'ancien, on efface tous les résultats de lecture en colonne, et par points de fixation.

Quand le CF grandit, on félicite le lecteur; On le motive en mettant en relief ses progrès par rapport à son plus petit CF jamais calculé.

Quand le CF a rapetissé, on montre au lecteur qu'il a déja fait mieux en affichant son meilleur CF jamais calculé.

S'IL N'AVAIT PLUS TRAVAILLE A LA LECTURE EN COLONNE OU PAR POINTS DE FIXATION DEPUIS PLUS DE 7 JOURS, ON FAIT LA CORRELATION AVEC SES MAUVAIS RESULTATS.

6.D.6) Diagnostics comparatifs

Il faut comparer les résultats de cette évaluation, avec ceux de la lecture en colonne et par points de fixation.

VOCABULAIRE :

On dira que les temps de lecture en colonne ou par points de fixation montrent une tendance à la baisse, si la moyenne de LC est sous l'intervalle de confiance du TMF, ou si la moyenne de LL est dans la partie A de l'intervalle décrit dans le schéma 2 pour la lecture en colonne, ou décrit dans le schéma 4 pour la lecture par points de fixation. (Cfr. infra)

On dira que les temps moyens de lecture en colonne montrent une tendance à la hausse, si la moyenne de LC est supérieur à l'intervalle de confiance du TMF.

On dira que les temps de lecture par points de fixation montrent une tendance à la hausse, si la moyenne de LC est supérieur à max+ dans le schéma 3 (cfr. infra).

REACTIONS :

Si depuis la dernière évaluation, le lecteur n'a pas fait assez l'un ou l'autre des 2 exercices de gammes au point que la moyenne des résultats n'est pas significative, il faut signaler qu'il y doit travailler plus souvent et régulièrement.

Si le CF a grandit et que la tendance montrée par un exercice est neutre ou la baisse, signaler au lecteur qu'il est trop pessimiste ;il doit se forcer à lire plus vite que ça vitesse confortable.

Si le CF a rapetissé et que la tendance montrée par un exercice est neutre ou à la hausse, signaler au lecteur qu'il est trop optimiste ;il doit se forcer à faire plus attention quand il lit vite.

S'IL N'AVAIT PLUS TRAVAILLE A LA LECTURE EN COLONNE OU PAR POINTS DE FIXATION DEPUIS PLUS DE 7 JOURS, ON DIT AU LECTEUR QUE LORSQU'IL ESPACE TROP SES SESSIONS, LES RESULTATS QUE LE DIDACTICIEL CONSERVE, DEVIENNENT CADUCS.

6.E) RECONNAISSANCE DE MOTS AU CONTOUR

6.E.1) Buts

- Améliorer la capacité à reconnaître les mots au contour seulement.
- Repérer les progrès.

6.E.2) Déroulement et présentation

- D'abord apparaît le modèle de mot à repérer au contour. Le lecteur tape la barre d'espacement, et le modèle disparaît et est repris 2 fois dans les lignes qui s'affichent.
- Dés qu'il a repéré les deux occurences, le lecteur tape la barre d'espacement et un nouveau modèle apparait.
- Le chrono se déclenche dés que disparaît le premier modèle et s'éteint après le repèrage du dernier modèle dans les lignes.
- Il y a 50 modèles en tout.

6.E.3) Valeurs retenues

- Liste des 10 dernièrs temps retenus. **(ABRV : LR)**
- Le meilleur temps jamais enregistré et le plus mauvais.
- Date de la dernière session, durant laquelle on a fait l'exercice.

REMARQUES :
- A l'ajout du 11 ième élément en début de liste, le premier est éliminé.
- Il faut que la liste contienne au moins 3 éléments pour que sa moyenne soit significative.
- L'avantage à ne retenir que les 10 dernières valeurs, est qu'on conserve une vue récente des capacités du lecteur

- Nombre de temps contenus dans LR.

6.E.4) Idées principales de diagnostic

- Si le lecteur effectue correctement l'exercice en moins de 2 minutes, il a une bonne capacité à reconnaître des mots au contour.

- Pour plus de 2 min 30, il y a de gros effort à faire.

6.E.5) Diagnostic faisant suite à l'exercice

Soit T, le temps mis pour cet exercice.

T est dit absurde, s'il est plus petit que 1 min 45, ou plus grand que 4 minutes.

On affiche toujours T, la moyenne de LR, le meilleur temps jamais enregistré. On signale toujours si la moyenne est significative, et si T est plus le long temps jamais enregistré. Si le lecteur n'a pas travaillé depuis 7 jours, on l'indique.

1- Si T est absurde, on le fait remarquer. Rien n'est mémorisé.

2- Si T est non absurde mais supérieur à 2 min 30,

T est inséré dans LR.

Il faut signaler au lecteur qu'il doit se forcer à descendre au moins sous les 2 minutes 30.

On compare T à sa moyenne.

Si le lecteur n'avait plus fait d'exercices depuis 7 jours, il faut signaler qu'il lirait mieux s'il s'excerçait plus souvent.

3- Si T se situe entre 2 min et 2 min 30,

T est inséré dans LR.

Il faut signaler au lecteur qu'il doit descendre à 2 minutes

On compare T à sa moyenne.

Si T est plus petit que sa moyenne, et si le lecteur n'avait plus fait d'exercices depuis 7 jours, il faut signaler qu'il lirait mieux s'il s'excerçait plus souvent.

4- Si T est plus petit ou égal à 2 min,

T est inséré dans LR.
Il faut dire au lecteur qu'il vient de montrer une bonne capacité à reconnaître les mots au contour.

On compare T à sa moyenne, si cette moyenne se au-dessus des 2 minutes , on félicite le lecteur pour ce progrès, et on lui propose de recommencer pour voir confirmer. Mais si cette moyenne dépassait 2 min 30, on fait remarquer que sa performance est étrangement bonne par rapport à sa moyenne.

REMARQUE : Comme il faut moins de temps pour repérer un mot à son contour, que pour le lire en un point de fixation, on ne peut pas comparer un temps de reconnaissance d'un mot au contour avec le TMF étalon. Nous n'avons donc plus de valeur de référence pour contrôler si le lecteur a repéré les mots correctement. Par conséquent on est obligé de faire confiance au lecteur.

6.E.6) Diagnostic général

On affiche toujours la moyenne de LR, le meilleur temps jamais enregistré. On signale toujours si la moyenne est significative. Si le lecteur n'a pas travaillé depuis 7 jours, on l'indique. Pour l' encourager, on compare toujours la moyenne au plus long temps jamais enregistré .

1- Si la moyenne de LR est supérieur à 2 min 30,

On indique la mauvaise capacité à reconnaître les mots au contour. Il faut encore s'exercer. Si le meilleur temps est au moins plus petit que 2 min 30, on insistera sur le fait qu'il a déjà fait mieux.

Il faut signaler au lecteur qu'il doit se forcer à descendre au moins sous les 2 minutes 30.

Si le lecteur n'avait plus fait l'exercice depuis 7 jours, il faut signaler qu'il lirait mieux s'il s'excerçait plus souvent.

3- Si la moyenne de LR se situe entre 2 min et 2 min 30,

Il faut signaler au lecteur qu'il doit descendre à 2 minutes, qu'il peut descendre à son meilleur résultat.

4- Si la moyenne de LR est plus petite ou égale à 2 min,

le lecteur montre une bonne capacité à reconnaître les mots au contour. Il ne doit plus effectuer cet exercice que pour maintenir sa forme. On le félicite;

6.F) EVITER LES CONFUSIONS DE MOTS

6.F.1) Buts

- Améliorer la capacité à reconnaître un mot mélangé à d'autres qui lui ressemblent.

- Repérer les progrès.

6.F.2) Déroulement et présentation

- Dés que le lecteur tape sur la barre d'espacement, les premières phrases absurdes mais contenant des mots semblables, s'affichent jusqu'à remplir l'écran.

- Quand le lecteur régresse, il doit appuyer sur ENTER.

- Il tape sur la barre d'espacement quand il a lu toutes les phrases, ainsi d'autres s'affichent.

Il y a 140 phrases au total.

- Le chrono se déclenche à l'affichage des premières phrases, et s'arrête dés que le lecteur a lu la 140 ième.

REMARQUE : Appuyer ENTER est un peu gênant pour la concentration, mais ce l'est beaucoup moins que de cocher des phrases au crayon.

6.F.3) Valeurs retenues

- Liste des 10 dernièrs temps retenus, de leur nombre de régressions correspondantes. **(ABRV : LR)**

- Le temps du dernier exercice, et le nombre de régressions.

- Le meilleur résultat en temps et en nombre de régressions.

- Date de la dernière session, durant laquelle on a fait l'exercice.

REMARQUES :
- A l'ajout du 11 ième élément en début de liste, le premier est éliminé .
- Il faut que la liste contienne au moins 3 éléments pour que sa moyenne soit significative.
- L'avantage à ne retenir que les 10 dernières valeurs, est qu'on conserve une vue récente des capacités du lecteur

6.F.4) Idées principales de diagnostic

Il est très difficile d'automatiser le diagnostic, car F.Richaudeau donne spécialement peu d'informations. En effet, il ne dit pas ce qu'il faut faire quand l'exercice dure plus de 2 minutes et je n' ai pas d'éléments de comparaison avec d'autres exercices. Par conséquent, pour ne pas prendre de décisions par trop arbitraires, je laisserai souvent le diagnostic au soin du lecteur.

6.F.5) Diagnostic faisant suite à l'exercice

Soit T, le temps mis pour cet exercice.
Soit N, le nombre de régressions pour cet exercice.
T est dit absurde, s'il est plus petit que 1 min 45, ou plus grand que 4 minutes.

On affiche toujours T, la moyenne de LR, le meilleur temps jamais enregistré.

1- Si T est absurde, on le fait remarquer. Rien n'est mémorisé.

2- Si T est non absurde mais supérieur à 2 minutes

T est inséré dans LR.

On indique au lecteur qu'il doit descendre sous 2 minutes, pour que le didacticiel analyse ses réactions quand il lit vraiment vite. On ajoute que, pour un temps de 2 minutes, il ne faut pas plus de 3 régressions. Pour 4 à 8 régressions, le lecteur doit faire plus attention quand il lit vite. Pour plus de 8 régressions, il faut faire l'exercice jusqu'à descendre à 2 minutes avec un maximum de 3 régressions.
Si le meilleur résultat enregistré jusqu'à présent est plus grand que 2 minutes, T et N deviendront le meilleur résultat si T est plus petit.

On compare T à sa moyenne.

Si le lecteur n'avait plus fait d'exercices depuis 7 jours, et que la moyenne de LR dépasse 2 minutes, il faut signaler qu'il lirait mieux s' il s'excerçait plus souvent.

3- Si T est plus petit ou égal à 2 min

T est inséré dans LR.

On indique que, pour un temps de 2 minutes, il ne faut pas plus de 3 régressions. Pour 4 à 8 régressions, le lecteur doit faire plus attention quand il lit vite; Pour plus de 8 régressions, il faut faire l'exercice jusqu'à descendre à 2 minutes avec un maximum de 3 régressions.

Si le meilleur résultat enregistré jusqu'à présent est plus petit que 2 minutes, T et N deviendront le meilleur résultat si N est plus petit.

Si N ‹= 3 régressions, on lui dit qu'il n'est plus nécessaire de faire l'exercice. Et si la moyenne était déjà plus petite que 2 minutes avec au plus 3 régressions, on insite sur le fait qu'il n'est plus nécessaire de recommencer l'exercice.

Si N › 3, on compare T à sa moyenne.

6.F.6) Diagnostic général

On affiche toujours le dernier résultat, la moyenne de LR, le meilleur temps jamais enregistré.

Si le dernier exercice enregistré indique un temps inférieur à 2 minutes avec au plus 3 régressions, on dit qu'il n'est plus nécessaire de faire l'exercice.

1- Si la moyenne de LR est supérieure à 2 minutes

On INDIQUE que pour un temps de 2 minutes, il ne faut pas plus de 3 régressions; pour 4 à 8 régressions, le lecteur doit faire plus attention quand il lit vite; Pour plus de 8 régressions, il faut faire l'exercice jusqu'à descendre à 2 minutes avec un maximum de 3 régressions.

3- Si la moyenne de LR est plus petite ou égale à 2 minutes

On INDIQUE que, pour un temps de 2 minutes, il ne faut pas plus de 3 régressions; Pour 4 à 8 régressions, le lecteur doit faire plus attention quand il lit vite; Pour plus de 8 régressions, il faut faire l'exercice jusqu'à descendre à 2 minutes avec un maximum de 3 régressions.

6.G) EVALUATION DES TENDANCES A LIRE MOT A MOT

6.G.1) Buts

- Repérer les tendances à lire mot à mot.

6.G.2) Déroulement et présentation

- Dés que le lecteur tape sur la barre d'espacement, apparaît le texte en colonne. Il retape sur la barre d'espacement dés qu'il a fini de lire l'écran.

- Quand le texte en colonne est lu, le lecteur tape sur la barre d'espacement. Le texte normal apparaît. Il retape sur la barre d'espacement dés qu'il a fini de lire l'écran.

-Le chrono se déclenche dés qu'apparaissent les premiers mots en colonne, et s'éteint après la lecture du dernier.

- Le chrono se re-déclenche dés qu'apparait le début du texte normal, et s'arrête dés qu'il est lu.

6.G.3) Valeurs retenues

- Un booléen indiquant s' il y a tendance à lire mot à mot. Un booléen est une variable dont la valeur est **vrai ou faux**.

6.G.4) Diagnostic faisant suite à l'exercice

- Si le temps de lecture du texte en colonne n'est pas au moins deux fois plus petit que le temps de lecture normal, il y a tendance à lire mot à mot.

6.G.5) Réactions

Si on diagnostique qu'il n'y a pas de tendance pour la première fois, on félicite le lecteur.

Si on avait déjà diagnostiqué qu'il n'y a pas de tendance, on signale au lecteur qu'il ne faut pas faire ce test inutilement.

Pour que le lecteur perde cette tendance, on conseillera beaucoup d'exercices de lectures en colonne et par points de fixation.

6.G.6) Diagnostic général

- On indique s'il y a tendance.
- S'il y a tendance, 2 sessions après la dernière évaluation, on conseillera de reévaluer. Au début de la 3ième session suivant la dernière évaluation, on forcera la dernière reévaluation.
- S'il n'y a pas de tendance, on signale au lecteur qu'il n'est plus nécessaire de faire l'exercice.

6.H) EVALUATION DES TENDANCES A SUBVOCALISER

6.H.1) Buts

- Repérer les tendances à subvocaliser.

6.H.2) Déroulement et présentation

- Le lecteur tient son larynx avec son pouce, ou retient un mouchoir du bout des lèvres.

- Dés que le lecteur tape sur la barre d'espacement, apparaît une liste de phrases dont la lecture orale provoque des sons vibratoires.

- Le lecteur lit les phrases de plus en plus vite.

6.H.3) Valeurs retenues

- AUCUNE

6.H.4) Diagnostic faisant suite à l'exercice

REMARQUE : Le didacticiel n'apporte rien de mieux que le livre.

Si le lecteur sent bouger son larynx, ou si son mouchoir tombe, c'est qu'il a tendance à subvocaliser.

6.I) LECTURE NATURELLE

6.I.1) Buts

- Posséder une évaluation standard de la vitesse de lecture;

- Permettre de tester les vraies améliorations apportées par les exercices.

6.I.2) Déroulement et présentation

- Le lecteur choisit un texte d'un niveau de difficulté qui lui paraît être moyen et qu'il n'a pas encore lu.

- Dés que le lecteur tape sur la barre d'espacement, le texte apparaît .

- Il retape sur la barre d'espacement dés qu'il a fini de lire l'écran.

- Le chrono se déclenche dés qu'apparaît le début du texte, et s'arrête dés que le texte est lu.

- Enfin le lecteur répond aux questions affichées à l'écran.

6.I.3) Valeurs retenues

- Les 2 premiers résultats en nombre de signes à l'heure et en taux de bonnes réponses.

- Le meilleur temps tel que le taux est plus grand que 6/10, et le taux correspondant.

- Le meilleur taux, et la vitesse correspondante.

- Moyenne de toutes les mesures de vitesse et taux.

- Moyennes des temps pour des lecture donnant un taux de réponse $>$ 6/10.

- Dernier résultat en taux et vitesse.

6.I.4) Initialisation

Lors de la premières évaluation, le lecteur lira 2 textes. Si le taux est <= 6/10 à la première lecture, on proposera de lire plus lentement le second texte. Si au contraire, le taux est > 6/10, on proposera de lire plus vite.

Ensuite on calculera la moyenne des 2 vitesses et taux. Cela nous servira de valeur de référence de départ.

6.I.5) Diagnostic général

Par la suite, on se contentera d'afficher les valeurs retenues citées avant. Pour que le lecteur puisse se repérer, on affichera une échelle de vitesse allant du lecteur lent jusqu'au très rapide.

Je n'ai pas à conseiller d'aller plus vite ou moins vite pour telle ou telle lecture car le lecteur doit adapter sa vitesse à la difficulté du texte. C'est au lecteur de tirer les conclusions.

6.J) DES VALEURS A CHANGER.

Pour diminuer le coté arbitraire des décisions, un observateur humain peut considérer que certaines valeurs que j'utilise sont mal choisies; Il doit par conséquent pouvoir les changer.

Voici la liste d'éléments qui formeront ce que nous appelleront l'**ENVIRONNEMENT** du didacticiel :

- Les 35% de la règle.
  (Valeur max autorisée 80%)

- Nbre de valeurs à retenir avant que leur moyenne soit représentative du TMF.
  (Par défaut:100)

- Nbre maximum d'éléments dans les listes LL,LC LR.
  (Par défaut:10)

- Nbre d'éléments rendant la moyenne des valeurs de ces listes significatives.
  (Par défaut:3)

- Limites de temps absurdemment trop long ou trop court pour un TMF.
  (Par défaut: Min = 100 mls; Max = 400 mls)

- Limites de temps absurdemment trop long ou trop court pour la reconnaissance de 50 mots au contour.
  (Par défaut: Min = 1min 45 Max = 4 minutes)

- Limites de temps absurdemment trop long ou trop court pour la reconnaissance de 140 mots mélangés à d'autres qui leurs ressemblent.
  (Par défaut: Min = 1min 45 Max = 4 minutes)

- Limites de temps absurdemment trop long ou trop court en nombre de signes/heures.
  (Par défaut: Min = 40.000 signes Max = 510.000 signes.)

- Pourcentage du temps de lecture d'un texte normal, au dessus duquel le temps de lecture du même texte en colonne indique une tendance à lire mot à mot.
  (Par défaut: 50%)

- Imprécision admise dans l'évaluation du CF.
  (Par défaut:3)

- Largeur du pas en nombre de caractères, de l'élargissement à apporter aux mots ou demi-phrases, à chaque itération de l'exercice de lecture en colonne et par points de fixation;
  (Par défaut:3)

- Nombre de jours d'oisiveté après lesquels on estime que le lecteur ne travaille pas assez souvent;
  (Par défaut:7)

- Taux minimun de bonnes réponses révélateur d'une bonne compréhension lors d'une lecture naturelle.
(Par défaut: 7 sur 10)

- Taux de bonnes réponses en dessous duquel on diagnostique une mauvaise compréhension lors d'une lecture naturelle.
(Par défaut: 6 sur 10)

- Nombre maximum de régressions indiquant une bonne capacité à distinguer les mots lors de l'exercice servant à diagnostiquer la tendance à confondre des mots.
(par défaut:3)

- Nombre minimum de régressions (obtenu lors de l'exercice servant à diagnostiquer la tendance à confondre des mots) à partir duquel on conseille au lecteur de faire plus attention quand il lit vite.
(par défaut:4)

- Nombre minimum de régressions (obtenu lors de l'exercice servant à diagnostiquer la tendance à confondre des mots) à partir duquel on conseille au lecteur de refaire fréquemment l'exercice.
(par défaut:9)

CONCLUSIONS

Arrivé à la fin de ce travail, je remarque les points qui ont marqué ce mémoire.

Au départ, ce mémoire paraît simple, presque simpliste, tant le livre de F.Richaudeau est clair. Mais il a fallu vite déchanter.

Mon gros problème fut l'accoutumance à de nouvelles habitudes. Il m'a fallu de nombreux mois et les conseils de mon promoteur pour abandonner certains réflexes qui auraient étés fatals à la valeur du didacticiel.

Quand on a l'habitude de ne traiter que de problèmes où l'utilisateur n'a, à nos yeux, qu'une importance secondaire, il est bien dur de le faire passer à la première place.

La leçon la plus importante que je tire de ce mémoire, est que le traitement automatisé de problèmes humains nécessite un travail intérieur, invisible très important et laisse un gôut un peu amer. En effet, on est moins sûr que jamais que le logiciel fera ce qu'on désire. Il reste tellement dépendant de tous ses utilisateurs.

Mais peut-être est-ce là l'Informatique de l'avenir. Une Informatique moins contraignante, mais utilisée par des gens responsables. C'est aussi un choix que j'ai fait de libérer le lecteur. L'utilisation espérée du didacticiel nous renseignera sur la pertinence de ce choix. Seul un feed-back important pourrait vraiment assurer que les ambitions du mémoire sont rencontrées.

Le didacticiel facilite sûrement les manipulations, la concentration. La présentation des exercices n'est pas troublante et reste sobre. Bien sûr, on pourrait améliorer l'affichage des résultats et des diagnostics. Cela n'est pas un vrai problème.

Mais l'essentiel est de tester la viabilité d'un didacticiel d'aide à la lecture rapide. Si le prototype développé semble intéressant, il suffira de l'améliorer. J'ai toujours eut un souci de modularisation dans mes programmes . Cela devrait faciliter les modifications potentielles.

Il ne reste qu'à attendre. WAIT AND SEE !

BIBLIOGRAPHIE

Richaudeau (François),UNE METHODE MODERNE POUR APPRENDRE SANS PEINE LA LECTURE RAPIDE,
Marabout, Paris 1977.

L'auteur :
F.Richaudeau a écrit de nombreux ouvrages sur la lisibilité, l'écriture efficace et la pédagogogie. Il anime les éditions RETZ spécialisées dans les productions scolaires et para-scolaires.

Mansuet (François), LA LANGUE ECRITE.LA LECTURE, Fédération des Instituteurs Chrétiens, Bruxelles 1980.

L'auteur:
Manset est docteur en sciences pédagogiques.

AIDE-MEMOIRE

Voici la liste d'abréviations fréquemment utilisées dans ce mémoire :

TMF : **T**emps **M**oyen de **F**ixation.

C'est le temps moyen pendant lequel les yeux d'un lecteur se fixe sur un groupe de mots. (Voir p.17)

CF : **C**hamps de **F**ixation;

C'est la largeur maximale d'un groupe de mots qu'un lecteur peut reconnaître d'un coup. (Voir p.17)

LC : **L**iste des temps pour mots **C**ourts.

Type de liste de résultats utilisé pour l'évaluation des exercices de lecture en colonne et des exercices de lecture par points de fixation. (Voir p.58, p.63)

LL : **L**iste des temps pour mots **L**ongs.

Type de liste de résultats utilisé pour l'évaluation des exercices de lecture en colonne et des exercices de lecture par points de fixation. (Voir p.58, p.63)

LR : **L**iste des temps **R**etenus.

Type de liste de résultats utilisé pour l'évaluation des exercices de reconnaissance de mots au contour, des exercices pour éviter les confusions de mots.(Voir p.72,p.75)

FACULTES
UNIVERSITAIRES
N.D. DE LA PAIX
NAMUR

INSTITUT D'INFORMATIQUE

ANNEXE

Conception et implémentation d'un Didacticiel d'aide à LA LECTURE RAPIDE

**Mémoire de fin d'étude
en vue de l'obtention du grade de
LICENCIE ET MAITRE EN INFORMATIQUE**

Année académique 1986-1987

Auteur : Jacques Bournonville

Promoteur : Prof. Claude Cherton

RUE GRANDGAGNAGE, 21, B - 5000 NAMUR (BELGIUM)

TABLE DES MATIERES

GUIDE DE LECTURE

GENERALITES :

Ce sont quelques avertissements concernant l'implémentation.

PARAGRAPHE 1 :

Le didacticiel utilise de nombreux fichiers pour les exercices, les résultats, les diagnostics etc... etc...Afin de faciliter la compréhension de ma programmation, je définirai les noms de chacun de ces fichiers et leur utilité.

PARAGRAPHE 2 :

Pour limiter le nombre de fichiers, j'ai rangé un maximum d'informations dans chacun d'eux. Par exemple, les fichiers utiles pour la lecture naturelle, contiennent le texte, les questions, et les réponses possibles. Pour que la procédure qui traite ces fichiers puisse présenter les textes, poser les questions et réagir aux réponses données , il a fallu structurer ces fichiers de sorte à trouver la bonne information au bon endroit.

Ainsi dans ce paragraphe, nous détaillerons la structure des fichiers importants.

PARAGRAPHE 3 :

Nous expliquerons les grandes décisions à la base du didacticiel, ainsi que le contenu attendu de certaines disquettes.

PARAGRAPHE 4 :

Nous y expliquerons brièvement les fonctions que doit remplir le programme EXELECRA.

PARAGRAPHE 5 :

Nous y verrons la découpe modulaire du programme LECTRAPI.

GENERALITES

Pour cette implémentation nous distinguerons deux programmes:

- LECTRAPI : C'est le didacticiel proprement dit.
- EXELECRA : C'est le programme qui permet de créer des textes et des listes de mots utiles pour les exercices. Il permet aussi de modifier les variables d'environnement du didacticiel.

Etant donné que LECTRAPI est beaucoup plus important que EXELECRA, je porterai surtout mon intention sur LECTRAPI.

AVERTISSEMENTS

Au moment où j'écris ces lignes la programmation n'est pas terminée. Ainsi la composition des modules de la découpe modulaire de LECTRAPI est incomplète. Après la remise de mon mémoire, j'irai aussi loin que possible dans ce qu'il me reste à programmer.

Parce que c'est d'une importance secondaire, je ne ferai pas la découpe modulaire de EXLECRA. Il est problable que ce programme sera encore assez sommaire. Cependant les informations sur les fichiers contenues dans cette annexe, sont suffisantes pour permettre à quelqu'un d' en améliorer la programmation.

1)NOM DE FICHIER

1.A) FICHIER DE DIAGNOSTIC

Les procédures du module de diagnostic, produisent un fichier contenant les commentaires et diagnostics du didacticiel. Ce fichier porte le nom : "DIAG.DIA".

1.B) FICHIER CONTENANT LA THEORIE

Le didacticiel explique la théorie à la base de la lecture rapide dans le fichier "THEO".

1.C) FICHIERS DE COMMENTAIRES SUR LES EXERCICES

Pour chaque exercice, le lecteur peut accéder à des fichiers de commentaires sur :
a) L'utilité de l'exercice.
b) Le déroulement de l'exercice.
c) Une idée générale sur la façon dont le didacticiel diagnostique.

Les noms des fichiers seront:

LC : Pour la lecture en colonne.
PF : Pour la lecture par points de fixation.
EV : Pour l'évaluation du TMF et du CF.
MC : Pour la lecture de mots au contour.
CM : Pour la confusion de mots.
MM : Pour les tendances à lire mot à mot.
TS : Pour les tendances à subvocaliser.
LN : Pour la lecture naturelle.

Les suffixes sont :

.A : Pour les commentaires sur l'utilité.
.B : Pour les commentaires sur le déroulement.
.C : Pour les commentaires sur le diagnostic.

Exemple) Le fichier de commentaires sur l'utilité d'une lecture en colonne porte le nom : "LC.a".

1.D) FICHIER DE RESULTATS

Un fichier des résultats est associé à chaque utilisateur. Son nom est composé des 8 premières lettres du nom du lecteur, et du suffixe ".RES.

Exemple)
Soit un utilisateur répondant au nom de ALEXANDRE.
Le nom du fichier résultat associé est "ALEXANDR.RES".

1.E) FICHIERS CONTENANT DES LISTES DE MOTS

Pour la lecture des mots en colonnes, on a besoin de fichiers dont chaque élément est un mot ou une phrase courte d'une largeur précise.

Le nom de chaque fichier est "MOT". Le Suffixe de chaque fichier dépend de la largeur de ses éléments. Ainsi, le fichier contenant des mots d'une largeur plus petite ou égale à 6 caractères aura le préfixe ".L6". Pour des mots de 7 caractères, le suffixe sera .L7. Et ainsi de suite, jusqu'a un maximum de 26 caractères.

Exemple) Nom de fichier pour 15 caractères : "NOM.L15".

1.F) FICHIERS POUR LA LECTURE PAR POINTS DE FIXATION

Au départ, un préparateur entre un texte dans un fichier dont le suffixe est toujours : "NRM".

La procédure FILEMOT décompose le texte contenu dans ce fichier en ses mots.

Le résultat de cette opération est une liste de records dont le seul champ est un string de maximum 26 caractères. Cette liste est copiée dans un fichier dont le nom est celui du fichier source, auquel on donne le préfixe ".PF". Ce fichier destination a le même nom que le fichier origine mais avec un nouveau suffixe: ".PF".

A partir de ce fichier de suffixe ".PF", la procédure MAKEY crée un fichier de record dont le seul champs est un string de maximum 80 caractères. L'ensemble des strings de ce fichier forme un texte en Y, dont la largeur des demi-phrases dépend du CF du lecteur. Le nom du fichier dépend aussi de cette largeur. il sera composé du nom du fichier origine, du suffixe composé par ".Y" et la largeur.

Exemple)

Si le texte de départ se trouve dans le fichier "mémoire.NRM", le résultat du traitement par la procédure FILEMOT, est placé dans le fichier : "mémoire.PF". Et lors du traitement de ce dernier pour lecteur dont le CF = 14 caractères, MAKEY crée un fichier portant le nom :"mémoire.Y14".

1.G) FICHIERS POUR LA LECTURE NATURELLE

Le préparateur crée des fichiers contenant un texte et des questions sur ce texte. Ces fichiers ont obligatoirement le suffixe : ".LN".

1.H) FICHIERS POUR LES EXERCICES CONTRE LA LECTURE MOI A MOI

Le fichier utilisé pour la lecture mot à mot a un suffixe ".TND". Le texte qu'il contient peut être court. (20 lignes maximum). A partir de ce fichier, on doit en obtenir un autre qui contient la découpe mot à mot du texte. Ce résultat est sauvé dans un fichier de nom identique au fichier source mais ayant un suffixe ".MAM".

1.I) FICHIER POUR LES EXERCICES DE SUBVOCALISATION

Pour les exercices contre la confusion de mots, on a besoin d'un fichier de phrases auxquelles on peut accéder de façon aléatoire. Ce fichier s'appelle : "SUBVOC.PH"

1.J) FICHIER POUR LES EXERCICES CONTRE LA CONFUSION DE MOTS

Pour la confusion de mots, on a besoin d'un fichier de phrases auxquelles on peut accéder de façon aléatoire. Ce fichier s'appelle : "CONFUS.PH"

1.K) FICHIER POUR LA RECONNAISSANCE DE MOTS AU CONTOUR

Pour la reconnaissance de mots au contour, on a besoin d'un fichier de records auxquels on peut accéder de façon aléatoire. Ce fichier s'appelle : "CONTOUR.PH".

1.L) FICHIER CONTENANT LES VARIABLES D'ENVIRONNEMENT

Les variables contenant l'environnement du didacticiel peuvent appartenir à 2 sortes de fichiers.

- Soit l'utilisateur utilise l'environnement général commun contenu dans un fichier dont le nom est : "Général.ENV".

- Soit l'utilisateur a son propre environnement spécifique contenu dans un fichier dont le nom est composé des 8 premières lettres du nom de l'utilisateur, et dont le suffixe est : "ENV".

Exemple) Soit l'utilisateur répondant au nom de JEAN, le fichier portera le nom : "JEAN.ENV".

1.M) FICHIER CONTENANT LES COMMENTAIRES

Le fichier contenant toutes les phrases utilisées pour commenter le diagnostic se trouvent dans le fichier "TXT.DIA".

2) STRUCTURE DES FICHIERS

2.A) FICHIER DE RESULTATS

Les fichiers de suffixe .RES contiennent des informations sur le lecteur. Elles sont toutes de types REAL. Voici leur ordre et leur signification.

PARTIE FIXE:

1- Environnement : Si elle vaut 0, le lecteur a son propre environnement, sinon il utilise l'environnement général.

2- N° de la dernière session.
3- Nbre d'éléments dans la liste des TMF déja enregistrés.
4- Nombre d'éléments dans LC et LL pour la lecture en colonne .
5- Nombre d'éléments dans LC et LL pour la lecture par PF.
6- Nombre d'éléments dans LR pour la reconnaissance de mots au contour.
7- Nombre d'éléments dans LR pour l'exercice empêchant la confusion de mots.
8- Moyenne courante des TMF de la liste.
9- Le MX de l'équation d'acceptation.
10- Le MN de l'équation d'acceptation.
11- CF courant du lecteur
12- Le plus petit CF
13- Le plus grand CF
14- Meilleur moyenne de LL pour lecture en colonne.
15- Meilleur moyenne de LL pour lecture par PF.
16- Plus petit temps enregistré pour la la reconnaissance de mots au contour.
17- Plus grand temps enregistré pour la la reconnaissance de mots au contour.
18- Meilleur temps enregistré pour l'exercice empêchant la confusion de mots.
19- Nombre de régressions correspondant au 18.
20- Premier nombre de signes/heure retenu.
21- Second nombre de signes/heure retenu.
22- Taux de bonnes réponses pour 20.
23- Taux de bonnes réponses pour 21.
24- Nombre de signes/heure pour le meilleur taux > 0,6.
25- Taux de bonnes réponses pour 24.
26- Meilleur temps retenu en nombre de signes/heure.
27- Taux de bonnes réponses pour 26.
28- Moyenne de tous les nombres de signes/heure retenus.
29- Moyenne des taux correspondant au 29.
30- Moyenne des nombres de signes/heure pour un taux > 0,6.
31- Dernier nombre de signes/heure retenu.
32- Taux de bonnes réponses pour 31.
33- Variable indiquant s'il y a tendance ou pas.
(0 = tendance; 1 = Pas de tendance; 2 = Exercice pas encore fait une seule fois.)
34- N° de la session de la dernière évaluation des tendances à lire mot à mot.

35- Date de la dernière session. Exemple) pour le 29-9-1987 on multiplie l'année par 10000 à laquelle on ajoute le mois multiplié par 100, auquel on ajoute le jour. Cela donne 19870000 + 900 + 29 = 19870929.

36- Date de la dernière fois où le lecteur a fait l'exercice de lecture en colonne.
37- Date de la dernière fois où le lecteur a fait l'exercice de lecture par points de fixation.
38- Date de la dernière fois où le lecteur a fait l'exercice de reconnaissance de mots à la forme.
39- Date de la dernière fois où le lecteur a fait l'exercice pour éviter les confusions de mots.

PARTIE DYNAMIQUE

40- Liste des TMF retenus
41- LC pour lecture en colonne.
42- LL pour lecture en colonne.
43- Liste des numéros de session correspondant au 41.
44- LC pour lecture par points de fixation.
45- LL pour lecture par points de fixation.
46- Liste des numéros de session correspondant au 44.
47- LR pour reconnaissance de mots au contour.
48- LR pour confusion de mots.
49- Liste des nombres de régressions correspondant au 48.

2.B) FICHIER DE MOTS OU DE PHRASES

Tous les fichiers contenant des mots (suffixe ".PF", ".L5" à ".L6", ou "TXT.DIA" ou "CONFUS.PH" ou "SUBVOC.PH") sont des fichiers de records dont le seul champs est un string. (Longueur max = 26 pour .L5 à .L26; max = 80 pour les autres). Cela permet un accès aléatoire aux éléments du fichier. Cela ne serait pas possible avec un fichier texte.

2.C) FICHIER "CONTOUR.PH POUR LA RECONNAISANCE DE MOTS

C'est un fichier de records dont le premier champ est un string de 26 caractères maximum, et le second est un tableau de 3 strings de 80 caractères. Le premier champ contient le modèle qui doit se retrouver deux fois dans les strings du tableaux.

2.D) FICHIERS PRODUITS PAR LA PROCEDURE MAKEY

Ce sont les fichiers de suffixe ".Y5" à ".Y26".

A peu près la moitié des phrases d'un fichier de ce type comporte des demi-phrases d'une longueur égale au CF courant du lecteur. Pour l'autre moitié, les phrases sont d'une longueur égale au CF incrémentée d'un pas de 3 caractères. Le numéro de la première ligne de la seconde moitié des phrases se situe tout au début du fichier converti en string.

EX)

```
<Ligne 0>      '56'
<Ligne 1>       'bla bla bla' <1ère phrase de longueur normale>

     "    "    "    "    "
     "    "    "    "    "

<Ligne 55>      'bla bla bla' < Dernière phrase normale>
<Ligne 56>      'bla bla bla bla' < 1ère phrase plus longue>

     "    "    "    "    "
<FIN DU FICHIER>
```

2.E) FICHIERS UTILISES PAR LA PROCEDURE QUESTION

Ce sont les fichiers de suffixe ".LN".

Ils contiennent des textes à lire, les questions d'évaluation, et les informations utiles pour connaître les réponses proposées pour chaque question. Ils se composent chacun d'une ZONE-TEXTE, et d'autant de ZONE-QUESTION qu'il n'y a de questions d'évaluation.

La ZONE-TEXTE commence toujours par une ligne contenant un '#' directement suivi par au moins une autre ligne qui ne contient pas de '#'.

Une ZONE-QUESTION s'étend sur un maximun de 22 lignes.(la ligne du '#' étant exclue), et se termine :

- soit après 22 lignes .
- soit dés la rencontre d'un '#'.
- soit à la fin du fichier.

La ligne contenant un '#' de la zône-question, contient les réponses possibles à la question.

Le caratère non-blanc, collé à droite du '#', est la bonne réponse.

Exemple) #fgtr
f est la bonne réponse
gtr sont les autres réponses proposées.

Exemple d'une structure de fichier.)

```
#abc   (Début 1ère zône-question dont la bonne réponse est "a")
(1ère ligne affichée de la zône-question)


          (1ère ZONE-QUESTION)


( Dernière ligne affichée de la 1ère zône-question)
(Ligne perdue)
# av  (Ligne perdue car le "a" de "av" ne colle pas au '#')
#ght  ( Début 2nde zône-question dont la bonne réponse est 'g')


          ( 2nde ZONE-QUESTION )

(Dernière ligne de la 2nde zône question)
#sta  ( Début 3ième zône-question dont la bonne réponse est 's')

          ( 3ième ZONE-QUESTION )

(fin du fichier)
```

2.F) FICHIER D'ENVIRONNEMENT

Le fichier de suffixe ".ENV", contient toutes sortes de variables utiles pour le diagnostic. Ces variables sont de type REAL. Voici leur ordre dans le fichier :

1- Les 35% de la règle. (On retient 35)
(Valeur max autorisée 80%)

2- Nbre de valeurs à retenir avant que leur moyenne soit représentative du TMF.
(Par défaut:100)

3- Nbre maximum d'éléments dans les listes LL,LC LR.
(Par défaut:10)

4- Nbre d'éléments rendant la moyenne des valeurs de ces listes significatives.
(Par défaut:3)

5- Limites de temps absurdemment trop long ou trop court un TMF.
(Par défaut: Min = 100 mls; Max = 400 mls)

6- Limites de temps absurdemment trop long ou trop court pour la reconnaissance de 50 mots au contour.
(Par défaut: Min = 105 SEC Max = 240 SEC)

7- Limites de temps absurdemment trop long ou trop court pour la reconnaissance de 140 mots mélangés à d'autres qui leurs ressemblent.
(Par défaut: Min = 105 SEC Max = 240 SEC)

8- Limites de temps absurdemment trop long ou trop court en nombre de signes/heures.
(Par défaut: Min = 40.000 signes Max = 510.000 signes.)

9- Pourcentage du temps de lecture d'un texte normal, au dessus duquel le temps de lecture du même texte en colonne indique une tendance à lire mot à mot.
(Par défaut: 50%. On retient 50)

10- Imprécision admise dans l'évaluation du CF.
(Par défaut:3)

11- Largeur du pas en nombre de caractères, de l'élargissement à apporter aux mots ou demi-phrases, à chaque itération de l'exercice de lecture en colonne et par points de fixation;
(Par défaut:3)

12- Nombre de jours d'oisiveté après lesquels on estime que le lecteur ne travaille pas assez souvent;

13- Taux minimum de bonnes réponses révélateur d'une bonne compréhension lors d'une lecture naturelle.
(Par défaut: 7 sur 10)

14- Taux minimum de bonnes réponses en dessous duquel on diagnostique une mauvaise compréhension lors d'une lecture naturelle.
(Par défaut: 7 sur 10)

15- Nombre maximum de régressions indiquant une bonne capacité à distinguer les mots lors de l'exercice servant à diagnostiquer la tendance à confondre les mots.
(Par défaut: 3)

16- Nombre minimum de régressions (obtenu lors de l'exercice servant à diagnostiquer la tendance à confondre les mots) à partir duquel on conseille au lecteur de faire plus attention quand il lit vite.
(Par défaut: 4)

17- Nombre minimum de régressions (obtenu lors de l'exercice servant à diagnostiquer la tendance à confondre les mots) à partir duquel on conseille au lecteur de refaire fréquemment l'exercice.
(Par défaut: 9)

3) DECISIONS

3.A) DISQUE VIRTUEL

Vu nos grands besoins en traitement de texte, nous avons besoin d'une structure de taille variable, d'accès rapide, et permettant une grosse capacité de mémorisation.

La structure qui nous a parut la plus appropriée est le fichier. Cependant son accès est lent. Nous avons donc besoin d'un disque virtuel qui simule une disquette en mémoire centrale.

Le fichier a été préféré a la liste chainée pour des raisons de capacité de mémorisation, et de facilité d'implémentation. En effet les primitives de gestion de fichiers existent déjà.

Au lancement du programme le disque virtuel doit être vide.

3.B) FICHIERS OBLIGATOIRES

Certains fichiers doivent être accessibles fréquemment et rapidement. Par conséquent, lors de l'initialisation ils seront obligatoirement copiés sur le disque virtuel.

Ce sont - les fichiers de suffixe : ".A", ".B", ".C", ".L5" à ".L26"

- le fichier : THEO
- un fichier de suffixe ".ENV"
- le fichier TXT.DIA

3.C) FICHIERS POUR APPEL SYSTEME

Toutes les procédures utiles pour le didacticiel, et celles utiles à la fabrication des exercices sont trop nombreuses pour tenir dans un seul programme.

Il y aura donc le programme du didacticiel et celui de fabrication d'exercice.

Le premier se trouve dans le fichier "LECTRAPI.COM", le second dans le fichier "EXELCTRA.COM"

De plus le programme EXELCTRA.COM doit permettre un appel au traitement de texte de turbo.

Par conséquent, nous sommes contraint d'utiliser des fichiers systèmes de suffixe ".BAT", pour permettre à l'utilisateur de choisir entre deux programmes et pour permettre l'accès à turbo .

On ne pourra accéder au deux programmes qu'en réinitialisant le système. En effet, le fichier "AUTOEXEC.BAT" execute automatiquement les préliminaires nécessaires à l'exécution des programmes. (Création du disque virtuel, recopiage de fichiers, etc... etc...). Quand le lecteur aura choisi d'entrer de nouveaux exercices, il ne pourra accéder au programme du didacticiel qu'en réinitialisant le système . Et vice-versa...

3.D) CONTENU DES DISQUETTES

a) Disquette système.

Cette disquette contient :
- le système DOS
- le fichier AUTOEXEC.BAT
- le traitement de texte TURBO
- le fichier LECTRAPI.COM
- le fichier EXELECRA.COM

b) Disquette obligatoire.

- Tous les fichiers obligatoires.
- Le fichier contenant les résultats du lecteur.

c) Disquette d'exercices pour lecture par points de fixation.

Il y a deux étapes à la transformation des fichiers sources utilisés pour la lecture par points de fixation (Suffixe ".NRM")

La première qui a lieu dans le programme EXELECRA, durant laquelle le texte d'un fichier de suffixe ".NRM" est découpé en mots. Le résultat est déposé dans un fichier de suffixe ".PF". (Voir supra : 1.F LECTURE PAR POINTS DE FIXATION)

La seconde a lieu dans le programme LECTRAPI, durant laquelle les mots contenus dans le fichier de suffixe ".PF" sont agencés de façon à former un texte en Y adapté au CF du lecteur. Ce résultat est déposé dans un fichier de suffixe ".L5" à ".L26". Remarquons que cette transformation, qui demande un certain temps, a lieu juste avant la lecture, et que le lecteur peut faire l'exercice plusieurs fois avec le même texte.

PAR CONSEQUENT, il se peut que la version transformée d'un fichier de suffixe ".PF" existe déjà. Il serait donc ridicule de la recommencer inutilement. Donc si les textes de suffixe ".PF" et leur transformation sont accessibles en même temps, on pourra éviter des transformations inutiles.

Si ON veut plus de rapidité, ON CONSEILLERA DONC DE SAUVER SUR LA MEME DISQUETTE, LES FICHIERS DE SUFFIXE ".PF" ET LEURS VERSIONS TRANSFORMEES.

Mais si on veut pouvoir accéder à un maximum de fichiers, ON NE SAUVERA PAS LES VERSIONS TRANSFORMEES SUR LA MEME DISQUETTE.

d) Disquette de texte.

Le lecteur peut choisir l'exercice qu'il veut lors d'une lecture naturelle. Par conséquent, pour augmenter son choix, on rangera un maximum de fichiers de suffixe ".LN" sur la même disquette.

On effectue assez rarement le test de tendance de lecture mot à mot. De plus les textes utiles à ce test sont courts; on peut donc ranger les fichiers de suffixe ".TND" et ".MAM" sur la même disquette.

3.E) VIE DE DIAG.DIA.

Le fichier "diag.dia" qui contient l'analyse commentée du didacticiel au sujet d'un ou des exercices, est détruit dès que le lecteur l'a parcouru entièrement. Il n'existe que sur le disque virtuel et n'est jamais sauvé.

3.F) MODIFICATIONS DE FICHIERS OBLIGATOIRES

Les fichiers obligatoires ne sauraient pas être modifiés à partir du programme LECTRAPI. Seul le programme EXELECRA peut les modifier.

4) STRUCTURE DES VARIABLES RESULTATS

4.A) LES LISTES DE RESULTATS

Il y a un tableau de pointeurs. Chaque élément du tableau pointe vers une liste chainée. Voici l'ordre et la signification de ces listes chainées.

1- Liste des TMF retenus
2- LC pour lecture en colonne.
3- LL pour lecture en colonne.
4- Liste des numéros de session correspondant au 2.
5- LC pour lecture par points de fixation.
6- LL pour lecture par points de fixation.
7- Liste des numéros de session correspondant au 5.
8- LR pour reconnaissance de mots au contour.
9- LR pour confusion de mots.
10- Liste des nombres de régressions correspondant au 9.

4.B) LES RESULTATS SIMPLES

Ils ont conservés dans un tableau de réels. En voici l'ordre et la signification.

1- N° de la dernière session.
2- Nbre d'éléments dans la liste des TMF déja enregistrés.
3- Nombre d'éléments dans LC et LL pour la lecture en colonne .
4- Nombre d'éléments dans LC et LL pour la lecture par PF.
5- Nombre d'éléments dans LR pour la reconnaissance de mots au contour.
6- Nombre d'éléments dans LR pour l'exercice emp@chant la confusion de mots.
7- Moyenne courante des TMF de la liste.
8- Le MX de l'équation d'acceptation.
9- Le MN de l'équation d'acceptation.
10- CF courant du lecteur
11- Le plus petit CF
12- Le plus grand CF
13- Meilleur moyenne de LL pour lecture en colonne.
14- Meilleur moyenne de LL pour lecture par PF.
15- Plus petit temps enregistré pour la la reconnaissance de mots au contour.
16- Plus grand temps enregistré pour la la reconnaissance de mots au contour.
17- Meilleur temps enregistré pour l'exercice emp@chant la confusion de mots.
18- Nombre de régressions correspondant au 17.
19- Premier nombre de signes/heure retenu.
20- Second nombre de signes/heure retenu.
21- Taux de bonne réponse pour 19.
22- Taux de bonne réponse pour 20.
23- Nombre de signes/heure pour le meilleur taux > 0,6.
24- Taux de bonnes réponses pour 23.
25- Meilleur temps retenu en nombre de signes/heure.
26- Taux de bonnes réponses pour 25.
27- Moyenne de tous les nombres de signes/heure retenus.
28- Moyenne des taux correspondant au 27.

29- Moyenne des nombres de signes/heure pour un taux > 0,6.
30- Dernier nombre de signes/heure retenu.
31- Taux de bonnes réponses pour 30.
32- Variable indiquant s'il y a tendance ou pas.
(0 = tendance; 1 = Pas de tendance; 2 = Exercice pas encore fait une seule fois.)
33- N° de la session de la dernière évaluation des tendances à lire mot à mot.

4.C) LES DATES

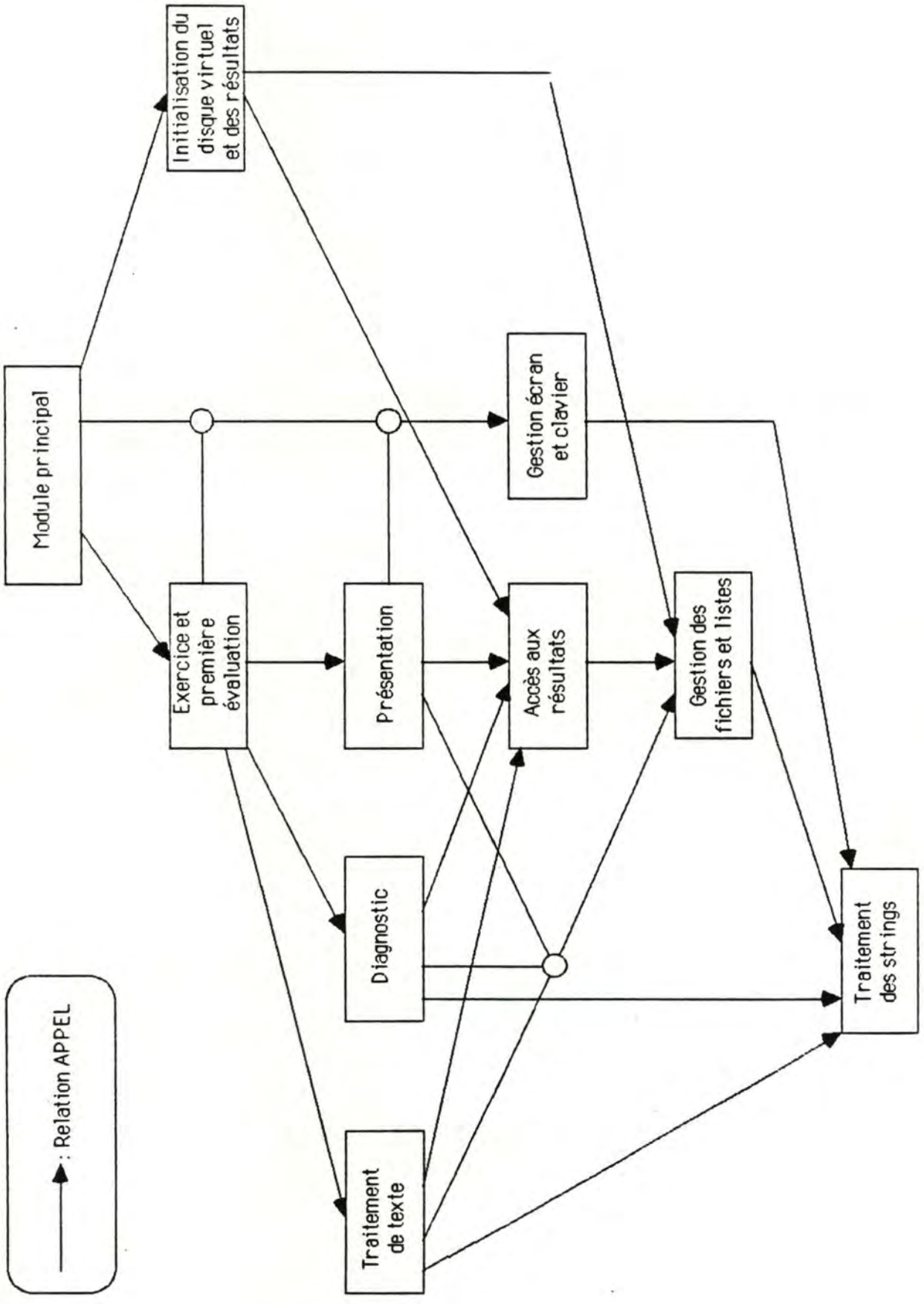
A chaque exercice, est associé la date de la dernière session durant laquelle l'exercice a été fait.

Elles sont rangées dans des tableaux de trois entiers signifiant le jour, mois, année.

1- Date de la dernière session.
2- Date de la dernière fois où le lecteur a fait l'exercice de lecture en colonne.
3- Date de la dernière fois où le lecteur a fait l'exercice de lecture par points de fixation.
4- Date de la dernière fois où le lecteur a fait l'exercice de reconnaissance de mots à la forme.
5- Date de la dernière fois où le lecteur a fait l'exercice pour éviter les confusions de mots.

5) DÉCOUPE MODULAIRE DU PROGRAMME LECTRAPI

5.4) SCHEMA



5.B) FONCTIONNALITE DES MODULES

PRECONDITION GENERALE

- Il existe un disque virtuel vide.
- Le même lecteur a suivi moins que 32767 sessions.

MODULE PRINCIPAL

FONCTIONALITES :

- Initialiser le disque virtuel.
- Initialiser les variables résultats.
- Expliquer la théorie à la base du didacticiel.
- Provoquer la première évaluation du lecteur.
- Proposer et diagnostiquer sur l'ensemble des exercices.
- Forcer les reévaluation du CF et des tendances à lire mot à mot.

MODULE INITIALISATION DU DISQUE VIRTUEL ET DES RESULTATS

FONCTIONNALITES :

- Copier les fichiers obligatoires sur le disque virtuel.
- Initialiser les variables de résultats et d'environnement.
- Sauver les résultats sur disquette.

MODULE EXERCICE ET PREMIERE EVALUATION

PRE-CONDITION :

- Il y a eut une initialisation du disque virtuel et des résultats.

FONCTIONNALITES :

- Expliquer l'utilité des exercices d'évaluation.
- Provoquer et diagnostiquer les 2 premières lectures naturelles.
- Provoquer la première évaluation des tendances à lire mot à mot.

- Provoquer la première évaluation du CF et du TMF.

- Provoquer la première évaluation des capacités à reconnaître un mot au contour.

- Permettre le choix entre différents exercices.

- Provoquer la réevaluation du CF et des tendances à lire mot à mot.

- Permettre le diagnostic sur base des résultats généraux de chaque exercice, et le diagnostic sur l'exercice qui vient d'être fait.

- Permettre l'affichage du diagnostics.

MODULE TRAITEMENT DE TEXTE

PRE-CONDITION :

- Il y a eut une initialisation du disque virtuel et des résultats.

FONCTIONNALITES :

- Préparer les fichiers et listes utilisés lors de l'exécution des exercices.

MODULE PRESENTATION

PRE-CONDITION :

- Il y a eut une initialisation du disque virtuel et des résultats.

FONCTIONNALITES :

- Permettre au lecteur d'exécuter les exercices.

- Calculer les résultats pour chaque exercice qui vient d'être effectué.

- Permettre l'affichage agréable à l'écran du contenu de fichiers texte.

MODULE DIAGNOSTIC

PRE-CONDITION :

- Il y a eut une initialisation du disque virtuel et des résultats.

FONCTIONNALITES :

- Evaluer un exercice qui vient d'être fait.

- Evaluer l'ensemble des résultats d'un type d'exercice.

- Comparer les résultats de la lecture en colonne avec ceux de la lecture par point de fixation.

- Générer un fichier de conseils et de commentaires sur le diagnostic.

MODULE GESTION ECRAN ET CLAVIER

FONCTIONNALITES :

- Afficher correctement à l'écran des menus, cadres, strings

- Gérer les entrées au clavier.

AVERTISSEMENT :

Ce module contient aussi les primitives de gestion de l'écran et du clavier proposées par le langage TURBO PASCAL.

Ce module contient les procédures SAVESCREEN et COPYSCREEN qui copie une zône de mémoire dans une autre.

MODULE ACCES AUX RESULTATS

PRE-CONDITION :

- Il y a eut une initialisation du disque virtuel et des résultats.

FONCTIONNALITES :

- Modifier, supprimer, ajouter, rendre des résultats.

- Rendre un code indiquant la position d'un temps par rapport à son intervalle de confiance.

- Rendre un code indiquant si il y a tendance à lire mot à mot.

MODULE GESTION LISTES ET FICHIERS

FONCTIONNALITES :

- Créer, nommer, mettre à jour, supprimer des fichiers.

- Créer, mettre à jour, supprimer des listes chainées.

AVERTISSEMENT :

Ce module contient aussi les primitives de gestion de liste et de fichiers proposées par le langage TURBO PASCAL.

MODULE TRAITEMENT DE STRINGS

FONCTIONNALITES :

- Modifier le contenus de strings.

AVERTISSEMENT :

Ce module contient aussi les primitives de gestion de strings proposées par le langage TURBO PASCAL.

5.C) EXPLICATIONS SUR CERTAINES RELATIONS ENIRE MODULES

Je me dois d'expliquer certains liens peu évidents entre certains modules.

a) Lien " TRT. DE TEXTE " et "ACCES AUX RESULTATS "

Ce lien est du à la nécessité d'adapter les exercices au niveau du lecteur; ainsi pour formatter un texte en Y ou pour créer une liste de mots en colonne, il est nécessaire d'accéder aux résultats pour connaître le CF du lecteur.

b) Lien " GESTION ET ECRAN ET CLAVIER " et "TRT. DE STRINGS"

Les procédures de gestion d'écran doivent souvent utiliser des strings qu'elles centrent, détruisent, blanchissent. Cela suppose l'utilisation de primitives de gestions de strings.

d) Lien "GESTION FICHIERS ET LISTES" et "INITIALISATION DISQUE VIRTUEL ET RESULTATS "

On initialise le disque virtuel en y copiant des fichiers, et beaucoup de résultats ont la forme de listes.

e) Lien "DIAGNOSTIC" et "GESTION FICHIERS ET LISTES "

Le module DIAGNOSTIC produit un fichier texte avec ses commentaires à partir du fichier des phrases contenues dans le fichier "TXT.DIA".

f) Lien "GESTION FICHIERS ET LISTES " et "TRT. DE STRINGS "

Le module de gestion de fichier doit former des noms de fichiers à partir d'autres noms de fichier; or ces noms sont conservés dans des strings.

g) Lien "DIAGNOSTIC" et "TRT. DE STRINGS"

Comme le module doit créer des commentaires à partir du fichier "TXT.DIA" et à partir de résultats. Pour former des phrases correctes, le module devra ajouter des valeurs de résultats aux phrases de "TXT.DIA".

Exemple) 'Ton TMF en millisecondes = ', 250.

250 sera transformé en string et ajouté à la phrase.

6) FONCTIONNALITES DE EXLECRA

Voici les fonctions que doit remplir EXLECRA :

6.A) POUR L'ENVIRONNEMENT

- Permettre la modification des fichiers d'environnement de suffixe ".ENV"

6.B) POUR LES EXPLICATIONS

- Permettre la modifications des fichiers contenant des explications sur les exercices, leur déroulement, leur utilité et sur la théorie qui en est la base. C'est à dire les fichiers de suffixe ".A", ".B" , ".C", et le fichier THEO.

6.C) POUR LES EXERCICES

a) Lecture en colonne

- Permettre la création des fichiers de mots de suffixe ".L5" à ".L26", soit interactivement élément par élément, soit à partir de la décomposition d'un texte normal, ou d'un texte tel que chaque ligne comporte un mot ou une phrase courte qui ne dépasse pas 26 caractères .

- Permettre la modification interactive de ces fichiers

- Permettre la suppression interactive de mots de ces fichiers

b) Lecture par points de fixation

- Permettre la création des fichiers de suffixe ".NRM" grâce à TURBO.

- Permettre la transformation d'un texte contenu dans les fichiers de suffixe ".NRM" en sa découpe en mots dans un fichier de suffixe ".PF". (La procédure FILEMOT qui effectue cette décomposition est déjà écrite et appartient au MODULE TRAITEMENT DE TEXTE du programme LECTRAFI. Cfr. Supra)

- Permettre le formattage des mots contenus dans les fichiers de suffixe ".PF" en un texte en Y rangé dans un fichier de suffixe allant de ".Y5" à ".Y26" selon le CF du lecteur. Remarquons que normalement ce formatage se déroule durant le programme LECTRAPI, juste avant que le lecteur ne fasse l'exercice. (La procédure MAKEY qui effectue cette décomposition est déjà écrite et appartient au MODULE TRAITEMENT DE TEXTE du programme LECTRAPI. Cfr.Supra)

c) Tendances à lire mot à mot

- Permettre la création des fichiers de suffixe ".TND" grâce à TURBO

- Permettre la transformation d'un texte contenu dans les fichiers de suffixe ".TND" en sa découpe en mots dans un fichier de suffixe ".MAM". (La procédure FILEMOT qui effectue cette décomposition est déjà écrite et appartient au MODULE TRAITEMENT DE TEXTE du programme LECTRAPI. Cfr. Supra)

d) Reconnaissance de mots au contour

- Permettre la création du fichier "CONTOUR.PH" soit interactivement soit à partir du fichier texte de suffixe "TXTCONT.CTX" lui-même créé à partir de TURBO.

- Permettre la modification interactive des éléments du fichier "CONTOUR.PH"

- Permettre la suppression interactive des éléments des fichiers de suffixe "CONTOUR.PH"

e) Confusion de mots

- Permettre la création du fichier "CONFUS.PH" soit interactivement, soit à partir du fichier texte "TXTCONF.CTX" lui-même crée à partir de TURBO.

- Permettre la modification interactive des éléments du fichier "CONFUS.PH".

- Permettre la suppression interactive des éléments du fichier "CONFUS.PH".

f) Tendances à subvocaliser

- Permettre la création du fichier "SUBVOC.PH" soit interactivement soit à partir du fichier texte "TXTSUBV.CTX" lui-même créé à partir de TURBO.

- Permettre la modification interactive des éléments du fichier "SUBVOC.PH".

- Permettre la suppression interactive des éléments du fichier "SUBVOC.PH".

g) Lecture naturelle

- Permettre la création de fichiers de suffixe ".LN" directement à partir de TURBO.

- Aider l'utilisateur à créer ces fichiers en respectant la structure. Il faudrait pour cela un petit traitement de texte plus approprié que TURBO et qui connaîtrait la structure des fichiers de suffixe ".LN".

- Permettre la suppression et la modification des fichiers de suffixe ".LN"

DECLARATION DE TYPES ET VARIABLES

```
CONST    cElTab = 4;
           E    = 3;        { cElTab - 1 }
         MxLgMot = 26;       { Longueur Max d'un mot.}
         MxLgPhr = 80;       { Longueur Max D'une phrase }
          MaxLgn = 25;
          MaxCol = 80;
            Incr = 5 ;       {Increment des mots pour lect. en col. et par P.F.}
            NbLn = 19;       { Pour Texte en Y, Nbre de lignes avant que
                               les demis-lignes se rejoignent ou s'eloignent de
                               nouveau.
                             }
      ColorSeg    = $B000;
      ColorOfs    = $8000;
      BWSeg       = $B000;
      BWOfs       = $0000;
      SizeScreen  = 7918;  { Taille en nombre de bytes .}

      MaxSimple   = 10;    { Nombre de r{sultats simples retenus}
      Maxdate     = 5 ;    { Nombre de dates retenues}
      Maxliste    = 5 ;    { Nombre de liste de r{sultats retenues}



TYPE        St6  =  string[6];
            st8  =  string[8];
            St26 =  string[26];
            St80 =  string[80];
          StMot  =  string[MxLgMot];
          StPhr  =  string[MxLgPhr];
        TabStMt  =  array[1..cElTab] OF stMot;
        Tabmot   =  array[1..16] OF StMot;
       FTabStMt  =  File OF TabStMt;
         PtStMt  =  ^RecStMt;    { Pointeur vers des MOTS. }
        RecStMt  =  RECORD
                       Elm  : StMot;
                       Nxt  : PtStMt;
                      Prev  : PtStMt;
                     Place  : ^integer;
                    END;

         PtStPh  =  ^RecStPh;    { Pointeur vers des PHRASES. }
        RecStPh  =  RECORD
                       Elm  : StPhr;
                       Nxt  : PtStPH;
                      Prev  : PtStPH;
                     Place  : ^integer;
                    END;

          PtInt  =  ^int;    { Pointeur des ENTIERS. }

            Int  =  RECORD
                       Val  : real;
                       Nxt  : Ptint ;
                      Prev  : Ptint;
                     Place  : ^integer;
                    END;

             PH3 = Array[1..3] OF StPhr;

          PtR35t = ^R35t;

            R35t = RECORD
                   Org : stMot;
```

```
                lgn : ph3;
                Nxt : PtR3sT;
               Prev : PtR3sT;
              Place : ^integer;

              END;


       TI = array[1..3] of integer;

            { ESC..Ord['z']  }
     Soa = Set OF 26..122;
    Mois = Set Of 1..12;


PtBckUp    = ^backUp;  { Tableau pour sauver Ecran }
BackUp     =  RECORD
              Ecran : array[1..SizeScreen] OF byte;
              nxt   : PtBckUp;
           END;

     rec = record
             elm : stPhr;
           END;
    frec = file of rec;


VAR   DebEcran     : Integer ABSOLUTE COLORSEG:Color0fs;
      vSimple      : array[1..MaxSimple] OF real; { Tableau des r{sultats
                                                    simples}
      vDate        : array [1..MaxDate] of ti; {Tableau de dates}
      vListe       : array [1..MaxListe] of ptint; { tableau de pointeurs
                                                     vers les listes }
```

MODULE TRAITEMENT DE STRINGS

```
procedure AllBlanc( var st : st80);

{ ARGT : ST
  RESLT: ST
 ACTION: En sortie ST est un string de MAXLGPHR blancs.
}
VAR   i : integer;

BEGIN
     st:= '';
     FOR i:= 1 to MxLgPhr do st:= st+ ' ';
END;

(* - - - - - - - - - - - - - fin de AllBlanc - - - - - - - - - - - - - - - - - - - - *)
procedure TrtCHIFFRE (var i, j, qte :integer; var s1, st : st80);

TYPE  Num = set of '0'..'9';
      car = set of char;

VAR   nm : num;
       c : car;
    cont : boolean;

BEGIN
     nm:= ['0'..'9'];      c:=['.',','];

     IF (i > 1) AND (i < length(st)) AND (st[i] IN c) AND
        (st[i-1] IN nm) AND (st[i+1] IN nm)
     THEN WHILE          ( i <= length(st))
                    AND ( (st[i] IN c) OR (st[i] IN nm) )
                    AND ( not( (st[i] IN c) AND (st[i-1] IN c) ) )
          DO BEGIN
             s1[j]:= st[i];
             i:= i+1;
             j:= j+1;
             qte:= qte+1;
          END;
END;
(* - - - - - - - - - fin de TrtCHIFFRE - - - - - - - - - - - - - - - - - - - - - *)
procedure KILLBLANC(var st : st80);

{ ARGT : ST
  RESLT: ST
 ACTION: En sortie ST a perdus tous ses caracteres blancs.
}

var i: integer;
BEGIN
    FOR i := 1 to length(st) DO IF (st[i] = ' ') THEN delete(st, i, 1);
END;
(* - - - - - - - - - - - - - - - - fin de KILLBLANC - - - - - - - - - - - - - - - - *)


Procedure CutBlanc ( var st : st80; last : integer);

{ PRE-COND : 0 < LAST <= lentgh(st)
  ARGT : ST, last
  RESLT: ST
 ACTION: En sortie ST a perdus tous les caracteres blancs compris entre
         le premier caractere non blanc a gauche de celui d'indice LAST et
         le caractere d'indice LAST.
}


BEGIN
     WHILE (last > 0) AND (st[last] = ' ') do
```

```
     BEGIN
          delete(st, last, 1);
          last:= last-1;
     END;
END;
(* - - - - - - - - - - - fin de CutBlanc - - - - - - - - - - - - - - - - - - *)
Procedure NxtWord( var st: st80; var ind : integer);

{ PRE-Cond : IND est un indice de ST
  ARGT     :  ST, ind >0
  RESLT    :  ind;
  ACTION   :  SI IND <> 0 alors IND est l'indice du caractere non-blanc suivant
              de ST.
}

BEGIN
     WHILE ( ind <= length(st)) AND (st[ind] = ' ') DO ind:= ind+1;

     IF ((ind > length(st)) OR (length(st)=0))  THEN ind:= 0;
END;
(*- - - - - - - - - -  - fin de NxtWord - - - - - - - - - - - - - - - - - - *)


Function CombienSigne(St:StPhr) : Integer;

{ CombienSigne = le nombre de caracteres non-blancs de ST.}

VAR i, j : integer;

BEGIN
     j:=0;
     FOR i:= 1 TO length(ST) DO IF (St[i] <> ' ') THEN j:=j+1;
     CombienSigne:= j;
END;

(*   - - - - - - - - -  FIN de CombienSigne - - - - - - - - - - - - - - - -*)

Procedure SsString(var st, result : st80; var depart, nxtw : integer;
                                                  longueur : integer);

       {Caract}res additionnels de fin de mot. '.', ',', ';', '*', '!', '?'
         Les mots sont s{par{s par des blancs.
         Un mot suivi de blancs puis des caract}res ci-dessous forment un mot.
         ex) "Jean  ." forme un mot, MAIS  " Jean.A " forme 2 mots "Jean."
              et "A".
              Les nombres "23.456" ou "4.000" forment 1 seul mot .
       }

TYPE  car = set of char;

VAR   ind,lg, i, j, qte : integer;
        cMoinsBlanc, c : car    ;
                    s2 : st80   ;
              oui, cont : boolean;

BEGIN
     { initialisations }

     lg:= length(st);       allblanc(result);    s2:= result;
     c:=[' ' , '.' , ',' , ';' , '*' , '!' , '?'];

     cMoinsBlanc:=['.' , ',' , ';' , '*' , '!' , '?'];
     cont:= true;

     { Trouver 1er caract}re du 1er mot  @ partir de DEPART. }
     Nxtword(st, depart);
```

```
IF (depart = 0)       { Si string blanc. }
THEN BEGIN
     nxtw:=0;
     cont:= false;
END
ELSE BEGIN
     i:= depart;
     j:= 1;
     qte:= 0;

     { Parcourir le mot jusqu'au 1er caract}re de fin de mot. }
     WHILE (i <= lg) AND NOT(st[i] IN c) DO
     BEGIN
          result[j]:= st[i];
          i:=i+1;
          j:=j+1;
          qte:= qte+1;
     END;
     TrtChiffre(i, j, qte, result, st);

     { Faire coller au mot les caract}res additionnels
       de fin de mot meme si ils sont s{par{s par des blancs.
     }
     WHILE (i <= lg) and (st[i] IN c) DO
     BEGIN
          result[j]:= st[i];
          i:=i+1;
          j:=j+1;
          qte:= qte+1;
     END;

    { Oter les caracteres suivant le mot copie }
     WHILE (j <= length(result)) do delete(result,j,1);

END;

     { A present, seul le premier mot ST est copi{ dans RESULT. }

{ Si on a epuise ST }
IF (cont) AND (i > lg) THEN nxtw:=0;

{ Si ST n'est pas epuise }
IF(cont) AND (i <= lg)
THEN IF (qte >= longueur)       { Le mot copi{ est assez long.}
     THEN BEGIN
          nxtw:= i;
          NxtWord(st, nxtw);
     END
     ELSE BEGIN                 { Le mot copie n'est pas assez long.}
          j:= longueur - qte;
          s2:= copy(st, i, j);

          oui:= false;

          IF (i+j <= lg)
          THEN IF NOT ( (st[i+j-1] IN c) AND (NOT(st[i+j] IN c)) )
               THEN BEGIN
                    ind:= i+j;
                    nxtword(st,ind);
                    oui:= (ind <> 0) AND (st[ind] in cMoinsBlanc);
                    oui:= (oui AND (st[i+j] = ' ')) or (st[i+j] <> ' ');
               END;

          IF (oui)
          THEN BEGIN
```

```
                    { Enlever les caracteres du mot tronque. }
                     j:= length(s2);

                     { D'abord oter les caracteres de fin de mot.
                       Utile pour le WHILE suivant, si s2[j] in c
                     }
                     WHILE (j > 0) AND (s2[j] IN c ) DO
                     BEGIN
                          delete(s2, j, 1);
                          j:= j-1;
                     END;

                     { Tuer le dernier caractere de S2
                       tant que l'on a pas epuise S2
                       et qu'on a pas rencontr{ un blanc ou le mot precedent
                     }
                     WHILE (j > 0) AND NOT(s2[j] IN c ) DO
                     BEGIN
                          delete(s2, j, 1);
                          j:= j-1;
                     END;
                END;

                cutblanc(s2, length(s2));
                result:= result + s2;
                nxtw:= i+ length(s2);
                NxtWord(st, nxtw);
           END;

     IF (cont) THEN cutblanc (result, length(result));

END;
(* - - - - - - - - - - fin de SsString - - - - - - - - - - - - - - - - - - - - *)
Procedure Kill5fx (Var nom : st26);

{
Argt   : nom
reslt  : nom
Action : Si NOM contient le caractere '.', KILL5FX supprime tous les caracteres
         de NOM @ partir du 1er caractere '.' rencontre.
}
BEGIN
     IF POS('.',nom) <> 0
     THEN delete(nom, Pos('.',nom), length(nom));
END;
{-  - - - -  - - - --  - - - - - -fin de Kill5fx - - - - - - - - - - - - - - - }
Procedure KillPfx (Var nom : st26);

{
Argt   : nom
reslt  : nom
Action : Si NOM contient le caractere ':', KILLPFX supprime tous les caracteres
         de NOM jusqu'@ inclus le 1er caractere ':' rencontre.
}
BEGIN
     IF POS(':',nom) <> 0
     THEN delete(nom, 1, Pos(':',nom));
END;
{-  - - - -  - - - --  - - - - - -fin de KillPfx - - - - - - - - - - - - - - - }
Procedure Add5fx (Var nom, sfx : st26 );


{
Argt   : nom, sfx
reslt  : nom
Action : Si NOM contient le caractere '.', ADD5FX supprime tous les caracteres
```

```
         de NOM @ partir du 1er caractere '.' rencontre.
         Et nom = nom + Sfx
}
BEGIN
     KillSfx(nom);
     nom:=nom+sfx;
END;
{-  - - - -  - - - --  - - - - - -fin de AddSfx - - - - - - - - - - - - - - }

Procedure AddPfx (Var nom, Pfx : st26 );
{
Argt   : nom, pfx
reslt  : nom
Action : Si NOM contient le caractere ':', ADDSFX supprime tous les caracteres
         de NOM jusqu'@ inclu le 1er caractere ':' rencontre.
         Et nom = Pfx + nom.
}
BEGIN
     KillPfx(nom);
     nom:= Pfx + nom;
END;
{-  - - - -  - - - --  - - - - - -fin de AddPfx - - - - - - - - - - - - - - }

procedure TrtNUM (var i:integer; var st : st80);

{ ARGT : i,st
  RESLT: St;
  ACTION : SI ST[i] = . ou , alors si ce sont des separateurs a l'interieur
           d'un nombre, I sera l'indice du dernier caracteres formant ce
           nombre.
           SINON I est inchange;

           EX) 4.000       2,56

TYPE  Num = set of '0'..'9';
      car = set of char;

VAR   nm : num;
       c : car;

BEGIN

     nm:= ['0'..'9'];     c:=['.',','];

     IF (i > 1) AND (i < length(st)) AND (st[i] IN c) AND
        (st[i-1] IN nm) AND (st[i+1] IN nm)
     THEN WHILE         ( i <= length(st))
                   AND ( (st[i] IN c) OR (st[i] IN nm) )
                   AND ( not( (st[i] IN c) AND (st[i-1] IN c) ) )
          DO BEGIN
             i:= i+1;
          END;
END;
(* - - - - - - - - - fin de TrtNUM - - - - - - - - - - - - - - - - - - - - - *)


Procedure StVersMots(var st, result : st80; var  cont : boolean);
{ Soit St = Un ensemble de mots : (MOT)'
  ALORS si ST est blanc ou vide => CONT = False
                                   RESULT = indefini;
        SINON => CONT = TRUE
                 RESULT = 1er mot de ST;
                 ST = (Mot)\ (1er mot)


Caract>res additionnels de fin de mot. '.', ',', ';', '*', '!', '?'
```

```
Les mots sont s{par{s par des blancs.
Un mot suivi de blancs puis des caract}res ci-dessous forment un mot.
       ex) "Jean  ." forme un mot, MAIS  " Jean.A " forme 2 mots "Jean."
            et "A".
            Les nombres "23.456" ou "4.000" forment 1 seul mot .
 }

TYPE  car = set of char;

VAR         depart, i   : integer;
          cMoinsBlanc, c : car    ;

BEGIN
     { initialisations }

     result:= '';
     cont:= true;
     c:=['  ' ,  '.' ,  ',' ,  ';' ,  '*' ,  '!' ,  '?'];

     cMoinsBlanc:=['.' ,  ',' ,  ';' ,  '*' ,  '!' ,  '?'];

     { Trouver 1er caract}re du 1er mot  @ partir de DEPART. }
     depart:= 1;
     Nxtword(st, depart);

     IF (depart = 0)       { Si string blanc. }
     THEN cont:= false
     ELSE BEGIN
          i:= depart-1;
          if ( i ) 0) THEN delete(st,1,i);
          i:=1;
     END;
     { Parcourir le mot jusqu'au 1er caract}re de fin de mot. }
     WHILE (i <= length(st)) AND NOT(st[i] IN c) DO i:=i+1;

     TrtNUM(i, st);

     { Faire coller au mot les caract}res additionnels
       de fin de mot meme si ils sont s{par{s par des blancs.
      }
     WHILE (i <= length(st)) and (st[i] IN c) DO
     BEGIN

          if (st[i] = ' ')
          THEN delete(st,i,1)
          ELSE i:=i+1;
     END;
     i:= i-1;
     result:= copy(st,1,i);
     delete(st,1,i);
END;
(* ---------------------- fin se StVersMots --------------------------- *)
Procedure MkLine( var Sg, Sd : stmot; var result : StPhr;  long : integer);

{  ARGT : Sg,Sd,long
   reslt: RESULT
  ACTION: RESULT sera un string, long de LONG caract}res, dont l'extreme
          gauche aura le contenu de SG, et l'extreme droite celui de SD.
          Le centre etant compose de "blancs".}

Var i : integer;
    bl: Stphr;

BEGIN
     i:= long - length(sg) - length(sd) ;
```

```
      IF (i < 1)
      THEN i:= 1;


      Bl:='                                                          ';
      bl:= bl + '                    ';
      result:= copy(bl,1,i);

      result:= Sg + result + sd;
END;
{ - - - - - - - -  - - - - --  fin de MkLine - - - - - - - - - - - - - - -  - - - - }
```

MODULE GESTION LISTES ET FICHIERS

```
Procedure Chain( Var Depart : PtStMT;   Qte: integer);

{ Si QTE > 0, alors Depart est le premier element de la chaine.
  HPT est Le pointeur vers le debut de la pile.
  Sinon DEPART = NIL, et HPT est indefini.}

VAR           i : integer;
    Courant, Pt : PtStMT;
           HPT : ^integer;

BEGIN
  IF (Qte <= 0)
  THEN  depart:= nil
  ELSE BEGIN
     Mark(HPT);
     New(Courant);
     courant^.nxt:= nil;
     courant^.Prev:= nil;
     courant^.Place:= HPT;
     Depart:= Courant;
     i:= 1;

     WHILE ( i < QTE) DO   { i = Nombre courant d'elements dans la liste.}
     BEGIN
          Mark(HPT);
          New(Pt);
          Pt^.Prev:= courant;
          Pt^.Nxt:=      nil;
          Pt^.Place:=    HPT;

          Courant^.nxt:=  Pt;
               Courant:=  Pt;

          i:=i+1;
     END;
  END;
END;
{ - - - - -  - - - - - -  fin de CHAIN - - - - -    - - - - - - - - - - - - }

procedure killchain ( VAR Depart :PtStMt);


{ Il doit exister une liste pointee par DEPART.}


VAR            Pt : PtStMt  ;
              HPT : ^integer;

BEGIN
     Pt:= Depart;

     { D'abord, se positionner en fin de liste. }


     WHILE (PT <> nil) DO
     BEGIN
          Pt:= Pt^.Nxt;
          IF (Pt <> nil) THEN Depart:= Pt;
     END;

     { A la sortie, DEPART pointe sur le dernier element de la liste.}

     WHILE ( Depart <> Nil) DO
     BEGIN
          HPT:= Depart^.place;
```

```
          Depart:= Depart^.prev;
          Dispose(HPT);
     END;
END;
(*- - - - - - - - - - - - fin de KillChain - - - - - -  - - - - - - - - - - - *)
Procedure NilPt(var d :ptstmt);

{ PRE-Cond : D NE POINTE PAS SUR L'ELEMENT D'UNE CHAINE.
  ACTION   : Effectue les initialisation necessaire a l'utilisation de
  la procedure AJELM Pour le pointeur D.}

BEGIN
     d:= nil;
END;

{ - - - - - - - - - - - fin de NilPt - - - - - - - - - - - - - - - - - - -}

Procedure CutChain( VAR From : PtStMt);

{ Si FROM <> NIL, CUTCHAIN detruit la chaine @ partir de l'element pointe
     par FROM.
  Sinon, pas d'effet.
}



BEGIN

     IF (From <> nil)
     THEN BEGIN
          IF (From^.prev <> Nil)
          THEN BEGIN
               From^.prev^.Nxt:= nil;
               From^.prev:= nil     ;
          END;
          killchain(from);
     END;
END;

(* - - - - - - - - -  - - fin de CutChain - - - - - - - - - - - - - - - - - -*)

procedure ReadElm( VAR More : boolean; VAR courant : PtStMt; Var st : StMot);

{ Si COURANT <> nil, alors ST contient le string pointe par COURANT,
 COURANT pointe sur l'element suivant Et More = TRUE. MORE vaut False
  s'il n'y a plus d'element suivant.
 Si Courant = Nil, ST est rendu vide et MORE = FALSE.
}

BEGIN
     IF (Courant = nil)
     THEN BEGIN
          st:='';
          MORE:= false;
     END
     ELSE BEGIN
          st:= courant^.elm;
          courant:= courant^.nxt;
          More :=(courant <> nil)  ;
     END;
END;
{ - - - - - - - - - - fin de ReadElm - - - - - - - - - - - - - - - - - - - - }

procedure AddElm( VAR More : boolean; VAR courant : PtStMt; st : StMot);

{ Si COURANT <> nil,  ST est ajoute dans la chaine a l'endroit pointe par
```

```
  COURANT. COURANT pointe sur l'element suivant Et More = TRUE.
  MORE vaut False, s'il n'y a plus d'element suivant.
  Si Courant = Nil, MORE = FALSE.
}

BEGIN
     IF (Courant = nil)
     THEN BEGIN
          st:='';
          MORE:= false;
     END
     ELSE BEGIN
          courant^.elm:= st;
          courant:= courant^.nxt;
          More :=(courant <> nil)  ;
     END;
END;
{ - - - - - - - - - - fin de AddElm - - - - - - - - - - - - - - - - - - - - - }

Procedure NxtPt( Var Courant : PtStMt);

{ PRE-CONDITIONS : Recoit COURANT <> nil,
   ET la procedure FILLTAB a rendu suivant = true;

  Recoit : Courant;
  Rend   : Courant;

  Le pointeur COURANT Pointe 16 elements plus loin dans la liste .
}

VAR i : integer;

BEGIN
     FOR i := 1 TO 16  DO courant:= courant^.nxt;
END;
(* - - - - - - - - - - - - -  fin de NxtPt - - - - - - - - - - - - - - - - - - *)

Procedure PrvPt( Var Courant : PtStMt);

{ PRE-CONDITIONS : Recoit COURANT <> nil,
  ET la procedure FILLTAB a rendu PRECEDENT = true;

  Recoit : Courant;
  Rend   : Courant;

  Le pointeur COURANT Pointe 16 elements avant dans la liste .
}

VAR i : integer;

BEGIN
     FOR i := 1 TO 16  DO courant:= courant^.prev;
END;
(* - - - - - - - - - - - - -  fin de PrvPt - - - - - - - - - - - - - - - - - - *)


Procedure FillTab (    courant: PtStMt ;
                   VAR tab    : TabMot ;
                   Var Qte    : Integer;
                   Var suivant, precedent : boolean
                  );

{ Recoit: Courant <>  NIL ;
  Rend  : Le reste;

    Si au depart COURANT <> nil, alors les QTE premiers mots de la liste
```

```
partant de COURANT sont copies dans TAB, dontt les elements vides seront
blancs.
Si SUIVANT est rendu = true, il a au moins dans la liste un element suivant
ceux copie
Si PRECEDENT est rendu = true, il a au moins dans la liste un element
precedent ceux copies;

)

VAR       more   :  Boolean;
            St   :  StMot;


BEGIN
     IF (Courant = NIL)
     THEN BEGIN
          Qte:= 0;
          suivant:= false;
          precedent:= false;
     END
     ELSE BEGIN

          qte:=1;
          WHILE (qte <= 16 )  DO
          BEGIN
               tab[Qte]:= '';
               qte:= qte+1;
          END;
          precedent:= (Courant^.prev <> nil);
          readElm( more, courant, st);
          qte:= 1;
          tab[qte]:= st;

          WHILE (Qte < 16) AND (more) DO
          BEGIN
               ReadElm( more, courant, st);
               qte:= qte + 1;
               tab[qte]:= st;
          END;

          suivant:=  more;
     END;
END;
(* - - - - - - - - - - - - fin de FillTab - - - - -  - - - -  - -- - - - - *)

Procedure AJElm(Var D : PtStMt; Var Val : StMot);

( ACTION : Cree un element de valeur VAL et l'ajoute a la liste pointee par
           D.  Si D = NIL on dit qu'il pointe sur une liste vide.
  Post-cond : D pointe sur le dernier element de valeur VAL, de la liste
              dont un element etait initialement pointe par D.
)


VAR PT :PtStMt;
    HPT : ^INTEGER;

BEGIN
     MARK(HPT);
     NEW(pt);
     Pt^.nxt:=nil;
     Pt^.place:=HPT;
     PT^.elm:= val;
     IF (D = NIL) THEN
     BEGIN
          D:=PT;
```

```
          D^.prev:= NIL;
     END
     ELSE BEGIN
          WHILE (D^.nxt <> NIL) DO D:= D^.nxt;
          PT^.prev:=d;
          D^.nxt:= pt;
          D:= pt ;
     END;
END;
{ - - - - - - - - - - - - -  fin de AjElm  - - - - - - - - - - - - - - - - - - }
Procedure InitList (Var D : PtStMt; Var Eol : Boolean);

{ Si en entree, D = nil c-a-d,pointe sur une liste vide eol = TRUE
  SINON en sortie D pointe sur le dernier element de la liste.
}

BEGIN
     EOL:= false;
     IF (D = NIL)
     THEN EOL:= TRUE
     ELSE WHILE (D^.PREV <> NIL) DO D:=D^.prev;
END;
{ - - - - - - - - - - - -  fin de InitList - - - - - - - - - - - - - - - - - - }

procedure NextLect(Var D : PtStMt; Var EOL : Boolean; Var Val : StMot);
{ Pre-cond : On est pas a la fin de la liste: eol = false ET/OU D <> NIL.
 Post-cond : D pointe sur l'element suivant s'il existe.
             La valeur de l'element courant est dans VAL.
}

BEGIN
     val:= D^.elm;
     IF (D^.nxt <> NIL)
     THEN D:= D^.nxt
     ELSE EOL:= true;
END;
{ - - - - - - - - - - - - - - -  fin de NextLect - - - - - - - - - - - - - - - }

Procedure ModifList ( i : integer; Var Val : stmot; Var Good : boolean;
                      d : ptstmt);
{ Pre-cond : D<>NIL;
     ARgts : i,val,d.
     reslt : good.
 Post-Cond : Si le Ieme Element existe, il prend la valeur VAL et
             GOOD = TRUE
             SINON Good = FALSE .
}


VAR
   eol : boolean;
       j: integer;

BEGIN
     InitList(d,EOL);
     good:= false;
     j:= 1;
     WHILE (J < i) AND ( D <> nil) DO
     BEGIN
          D:=D^.nxt;
          j:=j+1;
     END;
     IF (j=i) AND ( D <> NIL)
     THEN BEGIN
          D^.elm:= VAL;
          good:= true;
```

```
     END;
END;
{ - - - - - - - - - -  fin de ModifList  - - - - - - - - - - - - - - - - - -}
Procedure InList(d:ptStmt; Var val : StMot; Var Good : Boolean);

{ Pre-Cond : D pointe vers une chaine.
  Argts    : D, VAL
  reslt    : GOOD
  Post-cond: Si VAL appartient a la chaine pointee par D, alors GOOD  = true;
             SINON GOOD = False
}
VAR
   Gauche, Droite : ptStMt;

BEGIN
     good:= false;
     Gauche:= D;
     Droite:= D^.nxt;
     WHILE (NOT good) AND (Gauche <> NIL) DO
     BEGIN
          good:=(Gauche^.elm = Val);
          Gauche:=Gauche^.prev;
     END;
     WHILE (NOT good) AND (Droite <> NIL) DO
     BEGIN
          good:=(Droite^.elm = Val);
          Droite:= Droite^.nxt;
     END;
END;
{ ---------------------- fin de InList ---------------------------------- }




{ chaine de real}

procedure killchaiñI ( VAR Depart :PtInt);

{ Il doit exister une liste pointee par DEPART.}

VAR              Pt : Ptint  ;
                HPT : ^integer;

BEGIN
     Pt:= Depart;

     { D'abord, se positionner en fin de liste. }


     WHILE (PT <> nil) DO
     BEGIN
          Pt:= Pt^.Nxt;
          IF (Pt <> nil) THEN Depart:= Pt;
     END;

     { A la sortie, DEPART pointe sur le dernier element de la liste.}

     WHILE ( Depart <> Nil) DO
     BEGIN
          HPT:= Depart^.place;
          Depart:= Depart^.prev;
          Dispose(HPT);
```

```
     END;
END;
(*- - - - - - - - - - - - fin de KillChainI - - - - - - - - - - - - - - - - - - *)
Procedure NilPtI(var d :ptINT);

{ PRE-Cond : D NE POINTE PAS SUR L'ELEMENT D'UNE CHAINE.
  ACTION   : Effectue les initialisation necessaire a l'utilisation de
  la procedure AJELM Pour le pointeur D.}

BEGIN
     d:= nil;
END;

{ - - - - - - - - - - - fin de NilPtI - - - - - - - - - - - - - - - - - - -}


Procedure AJElmI(Var D : PtInt; Var Val : real);

{ ACTION : Cree un element de valeur VAL et l'ajoute a la liste pointee par
           D.  Si D = NIL on dit qu'il pointe sur une liste vide.
  Post-cond : D pointe sur le dernier element de valeur VAL, de la liste
              dont un element etait initialement pointe par D.
}


VAR PT :PtInt;
    HPT : ^INTEGER;

BEGIN
     MARK(HPT);
     NEW(pt);
     Pt^.nxt:=nil;
     Pt^.place:=HPT;
     PT^.val:= val;
     IF (D = NIL) THEN
     BEGIN
          D:=PT;
          D^.prev:= NIL;
     END
     ELSE BEGIN
          WHILE (D^.nxt <> NIL) DO D:= D^.nxt;
          PT^.prev:=d;
          D^.nxt:= pt;
          D:= pt ;
     END;
END;
{ - - - - - - - - - - - - fin de AjElmI - - - - - - - - - - - - - - - - - - }
Procedure InitListI (Var D : PtInt; Var Eol : Boolean);

{ Si en entree, D = nil c-a-d,pointe sur une liste vide eol = TRUE
  SINON en sortie D pointe sur le dernier element de la liste.
}

BEGIN
     EOL:= false;
     IF (D = NIL)
     THEN EOL:= TRUE
     ELSE WHILE (D^.PREV <> NIL) DO D:=D^.prev;
END;
{ - - - - - - - - - - - fin de InitListI - - - - - - - - - - - - - - - - - - }

procedure NextLectI(Var D : PtInt; Var EOL : Boolean; Var Val : real);
{ Pre-cond : On est pas a la fin de la liste: eol = false ET/OU D <> NIL.
 Post-cond : D pointe sur l'element suivant s'il existe.
             La valeur de l'element courant est dans VAL.
```

```
>

BEGIN
     val:= D^.val;
     IF (D^.nxt <> NIL)
     THEN D:= D^.nxt
     ELSE EOL:= true;
END;
{ - - - - - - - - - - - - - - -  fin de NextLectI - - - - - - - - - - - - - - - - >

Procedure ModifListI ( i : integer; Var Val : real; Var Good : boolean;
                       d : ptInt);
{ Pre-cond : D<>NIL;
     ARgts : i,val,d.
     reslt : good.
 Post-Cond : Si le Ieme Element existe, il prend la valeur VAL et
             GOOD = TRUE
             SINON Good = FALSE .
>


VAR
   eol : boolean;
     j: integer;

BEGIN
     InitListI(d,EOL);
     good:= false;
     j:= 1;
     WHILE (J < i) AND ( D <> nil) DO
     BEGIN
          D:=D^.nxt;
          j:=j+1;
     END;
     IF (j=i) AND ( D <> NIL)
     THEN BEGIN
          D^.val:= VAL;
          good:= true;
     END;
END;
{ - - - - - - - - - -  fin de ModifListI  - - - - - - - - - - - - - - - - ->


Procedure ShiftListI( D : PtInt; val : real);

{ Pre-Cond : D pointe sur un element d'une chaine.
  Argt     : D, Val;
  Post-cond: Soit une chaine dont un element est pointe par D :
                Tous les elements prennent la valeur de l'element precedent.
                La derniere valeur est perdue;  La premiere devient VAL.
>

BEGIN
     WHILE d^.nxt <> NIL DO D:= D^.nxt;
     WHILE D^.prev <> NIL DO
     BEGIN
          D^.val:=D^.prev^.val ;
          D:=D^.prev;
     END;
     D^.val:= val;
END;
{ -------------------------- fin de ShiftListI ---------------------------- >
```

```
{  Chaine de PHRASES }


procedure killchainP ( VAR Depart :PtStPh);

{ Il doit exister une liste pointee par DEPART.}

VAR             Pt : PtStPh  ;
               HPT : ^integer;

BEGIN
     Pt:= Depart;

     { D'abord, se positionner en fin de liste. }


     WHILE (PT <> nil) DO
     BEGIN
          Pt:= Pt^.Nxt;
          IF (Pt <> nil) THEN Depart:= Pt;
     END;

     { A la sortie, DEPART pointe sur le dernier element de la liste.}

     WHILE ( Depart <> Nil) DO
     BEGIN
          HPT:= Depart^.place;
          Depart:= Depart^.prev;
          Dispose(HPT);
     END;
END;
(*- - - - - - - - - - - - fin de KillChainP - - - - - -  - - - - - - - - - - - *)
Procedure NilPtP(var d :ptstPh);

{ PRE-Cond : D NE POINTE PAS SUR L'ELEMENT D'UNE CHAINE.
  ACTION   : Effectue les initialisation necessaire a l'utilisation de
  la procedure AJELM Pour le pointeur D.}

BEGIN
     d:= nil;
END;

{ - - - - - - - - - - fin de NilPtP - - - - - - - - - - - - - -- - -}

Procedure AJElmP(Var D : PtStPh; Var Val : StPhr);

{ ACTION : Cree un element de valeur VAL et l'ajoute a la liste pointee par
           D.  Si D = NIL on dit qu'il pointe sur une liste vide.
  Post-cond : D pointe sur le dernier element de valeur VAL, de la liste
              dont un element etait initialement pointe par D.
}


VAR PT :PtStPh;
    HPT : ^INTEGER;

BEGIN
     MARK(HPT);
     NEW(pt);
     Pt^.nxt:=nil;
     Pt^.place:=HPT;
     PT^.elm:= val;
```

```
     IF (D = NIL) THEN
     BEGIN
          D:=PT;
          D^.prev:= NIL;
     END
     ELSE BEGIN
          WHILE (D^.nxt <> NIL) DO D:= D^.nxt;
          PT^.prev:=d;
          D^.nxt:= pt;
          D:= pt ;
     END;
END;
{ - - - - - - - - - - - -  fin de AjElmP  - - - - - - - - - - - - - - - - - - }
Procedure InitListP (Var D : PtStPh; Var Eol : Boolean);

{ Si en entree, D = nil c-a-d,pointe sur une liste vide eol = TRUE
  SINON en sortie D pointe sur le dernier element de la liste.
}

BEGIN
     EOL:= false;
     IF (D = NIL)
     THEN EOL:= TRUE
     ELSE WHILE (D^.PREV <> NIL) DO D:=D^.prev;
END;
{ - - - - - - - - - - -  fin de InitListP - - - - - - - - - - - - - - - - - - }

procedure NextLectP(Var D : PtStPh; Var EOL : Boolean; Var Val : StPhr);
{ Pre-cond : On est pas a la fin de la liste: eol = false ET/OU D <> NIL.
 Post-cond : D pointe sur l'element suivant s'il existe.
             La valeur de l'element courant est dans VAL.
}

BEGIN
     val:= D^.elm;
     IF (D^.nxt <> NIL)
     THEN D:= D^.nxt
     ELSE EOL:= true;
END;
{ - - - - - - - - - - - - - - -  fin de NextLectP - - - - - - - - - - - - - - - }

Procedure InListP(d:ptStPh; Var val : StPhr; Var Good : Boolean);

{ Pre-Cond : D pointe vers une chaine.
  Args     : D, VAL
  reslt    : GOOD
  Post-cond: Si VAL appartient a la chaine pointee par D, alors GOOD  = true;
             SINON GOOD = False
}
VAR
   Gauche, Droite : ptStPh;

BEGIN
     good:= false;
     Gauche:= D;
     Droite:= D^.nxt;
     WHILE (NOT good) AND (Gauche <> NIL) DO
     BEGIN
          good:=(Gauche^.elm = Val);
          Gauche:=Gauche^.prev;
     END;
     WHILE (NOT good) AND (Droite <> NIL) DO
     BEGIN
          good:=(Droite^.elm = Val);
          Droite:= Droite^.nxt;
     END;
```

```
END;
{ ---------------------- fin de InListP ---------------------------------- }



          { CHAINE DE RECORD DE 3 STRING + UN STRING MODELE }


procedure killchain3 ( VAR Depart :PtR3sT);

{ Il doit exister une liste pointee par DEPART.}

VAR             Pt : PtR3sT  ;
               HPT : ^integer;

BEGIN
     Pt:= Depart;

     { D'abord, se positionner en fin de liste. }


     WHILE (PT <> nil) DO
     BEGIN
          Pt:= Pt^.Nxt;
          IF (Pt <> nil) THEN Depart:= Pt;
     END;

     { A la sortie, DEPART pointe sur le dernier element de la liste.}

     WHILE ( Depart <> Nil) DO
     BEGIN
          HPT:= Depart^.place;
          Depart:= Depart^.prev;
          Dispose(HPT);
     END;
END;
(*- - - - - - - - - - - - fin de KillChain3 - - - - - -  - - - - - - - - - - - - *)
Procedure NilPt3(var d :ptR3sT);

{ PRE-Cond : D NE POINTE PAS SUR L'ELEMENT D'UNE CHAINE.
  ACTION   : Effectue les initialisation necessaire a l'utilisation de
  la procedure AJELM Pour le pointeur D.}

BEGIN
     d:= nil;
END;

{ - - - - - - - - - - - fin de NilPt3 - - - - - - - - - - - - - - - - -- - -}

Procedure AJElm3(Var D : PtR3sT ;Var Val : Ph3; var val2 : stMot);

{ ACTION : Cree un element de valeur VAL et l'ajoute a la liste pointee par
           D.  Si D = NIL on dit qu'il pointe sur une liste vide.
  Post-cond : D pointe sur le dernier element de valeur VAL, de la liste
              dont un element etait initialement pointe par D.
}


VAR PT :PtR3sT;
    HPT : ^INTEGER;
```

```
BEGIN
     MARK(HPT);
     NEW(pt);
     Pt^.nxt:=nil;
     Pt^.place:=HPT;
     PT^.lgn:= val;
     PT^.org:= val2;
     IF (D = NIL) THEN
     BEGIN
          D:=PT;
          D^.prev:= NIL;
     END
     ELSE BEGIN
          WHILE (D^.nxt <> NIL) DO D:= D^.nxt;
          PT^.prev:=d;
          D^.nxt:= pt;
          D:= pt ;
     END;
END;
{ - - - - - - - - - - - - -  fin de AjElm3  - - - - - - - - - - - - - - - - - - }

procedure NextLect3(Var D : PtR3st; Var EOL : Boolean; Var Val : R3sT);
{ Pre-cond : On est pas a la fin de la liste: eol = false ET/OU D <> NIL.
 Post-cond : D pointe sur l'element suivant s'il existe.
             La valeur de l'element courant est dans VAL.
}

BEGIN
     val.org:= D^.org;
     val.lgn:= D^.lgn;
     IF (D^.nxt <> NIL)
     THEN D:= D^.nxt
     ELSE EOL:= true;
END;
{ - - - - - - - - - - - - - - -  fin de NextLect3 - - - - - - - - - - - - - - - }
Procedure InitList3 (Var D : PtR3sT ;Var Eol : Boolean);

{ Si en entree, D = nil c-a-d,pointe sur une liste vide eol = TRUE
  SINON en sortie D pointe sur le dernier element de la liste.
}

BEGIN
     EOL:= false;
     IF (D = NIL)
     THEN EOL:= TRUE
     ELSE WHILE (D^.PREV <> NIL) DO D:=D^.prev;
END;
{ - - - - - - - - - - - -  fin de InitList3 - - - - - - - - - - - -  - - - - - }
Procedure CopyF(VAR io,num : integer; sourceName,DestinationName : stmot);
{ Pre-cond : SourceName est le nom d'un fichier (PAS UN TEXTE) a copier
             dans le ficHier de nom DestnationName;
             Ces DEUX FICHIERS SONT FERMES.
  ARGTs     : SOURCEnAME, DestinationName;
  Reslt     : num, io
  Post-cond: SI IO = 0, Alors Le contenu du fichier dont le nom est dans
             SOURCENAME, est copie dans celui dont le nom est dans
             DestinationName.
             SI IO <> 0 alors IO vaut le code des entrees/ sorties car
             il y a eut un probleme avec le fichier Source si NUM=1, ou
             il y a eut un probleme avec le fichier Destination si NUM=2

ATTENTION: LA COPIE SE FAISANT PAR BLOC IL ARRIVE QUE DES ELEMENTS
           PARASITES S'AJOUTE A LA FIN DU FICHIER DESTINATION

}
```

```
CONST
   BufSize = 200;
   BufByteSize = 25600;

VAR
   Source,  Destination          : File                          ;

   Buffer                        : array[1..BufByteSize] OF Byte;
   NoOfRecToRead, Remaining      : integer                       ;


BEGIN
     assign(Source, sourceName);
     {$I-}  reset(source); {$I+}
     io:=ioresult;
     num:=1;
     IF (io = 0)
     THEN BEGIN
          assign(DESTINATION, DestinationName);
          {$I-}  rewrite (Destination); {$I+}
          io:=ioresult;
          num:=2;
     END;

     IF (io = 0) THEN
     BEGIN
          Remaining:= FileSize(source);
          WHILE (Remaining > 0) AND (io = 0) DO
          BEGIN
               IF (BufSize <= Remaining)
               THEN NoOfRecToRead:= BufSize
               ELSE NoOfRecToRead:= Remaining;

               {$I-}   BlockRead(Source, Buffer, NoOfRecToRead);       {$I+}
               io := ioresult;
               num:=1;

               IF (io = 0) THEN
               BEGIN
                    {$I-}   Blockwrite(Destination, buffer, NoOfRecToRead); {$I+
}

                    io:= ioresult;
                    num:=2;
                    Remaining:= Remaining - NoOfRecToRead;
               END
          END;
          close(source);
          close(destination);
     END;
END;
{ ---------------------------------- fin de copyF ------------------ }


Procedure copyT(VAR io,num : integer; sourceName,DestinationName : stmot);

{ Pre-cond : SourceName est le nom d'un fichier TEXTE a copier
             dans le ficHier de nom DestnationName;
             Ces DEUX FICHIERS SONT FERMES.
  ARGTs     : SOURCEnAME, DestinationName;
  Reslt     : num, io
  Post-cond: SI IO = 0, Alors Le contenu du fichier dont le nom est dans
             SOURCENAME, est copie dans celui dont le nom est dans
             DestinationName.
             SI IO <> 0 alors IO vaut le code des entrees/ sorties car
```

```
            il y a eut un probleme avec le fichier Source si NUM=1, ou
            il y a eut un probleme avec le fichier Destination si NUM=2


}


VAR
   Source,  Destination          : TEXT                              ;
   Ligne                         : string[255]                       ;

BEGIN
     assign(Source, sourceName);
     {$I-}  reset(source); {$I+}
     io:=ioresult;
     num:=1;
     IF (io = 0)
     THEN BEGIN
          assign(Destination, DestinationName);
          {$I-}  rewrite (Destination); {$I+}
          io:=ioresult;
          num:=2;
     END;

     IF (io = 0) THEN
     BEGIN
          WHILE (not (eof(source))) AND (io = 0) DO
          BEGIN
               {$I-} readln(source, ligne);   {$I+}
               io := ioresult;
               num:=1;

               IF (io = 0) THEN
               BEGIN
                    {$I-} writeln(destination, ligne); {$I+}
                    io:= ioresult;
                    num:= 2;
               END;
          END;
          close(source);
          close(destination);
     END;
END;
{ ------------------------ fin de CopyT ------------------------------ }

PROCEDURE LectDir( VAR Depart : PtStMt;  VAR Vide : boolean; Suffixe : StMot;
                       Dir    : StMot);

(* LECTURE DU DIRECTORY DE LA DISQUETTE QUI DOIT SE TROUVER SUR
   LE LECTEUR COURANT, LE CONTENU EST MIS DANS La liste pointe par depart.
   SEUL LES FICHIERS dir????????Suffixe SONT LUS
   SI SUFFIXE = .???, tous les fichiers de la DIR sont pris. *)

{
       This is a simple program to list out the directory of the
       (logged) drive DIR.
}
CONST
  CnbDir = 112;    { Nombre Maximum de fichiers sur une disquette }

TYPE
  RegRec =
    record
      AX, BX, CX, DX, BP, SI, DI, DS, ES, Flags : Integer;
    end;
```

```
VAR
  Courant                : PtStMt;
  More                   : Boolean;
  Regs                   : RegRec;
  DTA                    : array [ 1..43 ] of Byte;
  Mask                   : Array[1..14] Of Char;
  NamR                   : String[20];
  Error, I , position    : Integer;
                   st    : StMot;
begin

    WHILE (length(suffixe) < 4) DO SUFFIXE:= suffixe+ ' ';

    FillChar(DTA,SizeOf(DTA),0);          { Initialize the DTA buffer }
    FillChar(Mask,SizeOf(Mask),0);        { Initialize the mask }
    FillChar(NamR,SizeOf(NamR),0);        { Initialize the file name }

    Regs.AX := $1A00;             { Function used to set the DTA }
    Regs.DS := Seg(DTA);          { store the parameter segment in DS }
    Regs.DX := Ofs(DTA);          {   "    "       "     offset in DX }
    MSDos(Regs);                  { Set DTA location }
    Error := 0;
    for i:= 1 TO 2 do mask[i]:= dir[i];
    FOR i:= 3 to  10 DO MASK[i]:= '?';
    FOR i:= 1 to 4 DO mask[10+i] := suffixe[i]; { Use global search }
    Regs.AX := $4E00;                    { Get first directory entry }
    Regs.DS := Seg(Mask);                { Point to the file Mask }
    Regs.DX := Ofs(Mask);
    Regs.CX := 22;                { Store the option }
    MSDos(Regs);                  { Execute MSDos call }
    Error := Regs.AX and $FF;     { Get Error return }

    I := 1;                       { initialize 'I' to the first element }
    if (Error = 0) then
    repeat
      NamR[I] := Chr(Mem[Seg(DTA):Ofs(DTA)+29+I]);
      I := I + 1;
    until not (NamR[I-1] in [' '..'~']) or (I>20);

    NamR[0] := Chr(I-1);            { set string length because assigning }
                                    { by element does not set length }

    courant:= nil;
    VIDE:= true;
    if (Error = 0)
    then begin
           VIDE:= false;
           Chain(Depart, Cnbdir);
           courant:= depart;
           position := pos('.',NamR);
           IF ( Position <> 0) Then delete(NamR,position,4);
           st:=dir + NamR;
           AddElm(more, Courant, St);
    end;


    while (Error = 0) AND (More) do
    begin
      Error := 0;
      Regs.AX := $4F00;             { Function used to get the next }
                                    { directory entry }
      Regs.CX := 22;                { Set the file option }
      MSDos( Regs );                { Call MSDos }
      Error := Regs.AX and $FF;     { get the Error return }
      I := 1;
      repeat
```

L.S

```
        NamR[I] := Chr(Mem[Seg(DTA):Ofs(DTA)+29+I]);
        I := I + 1;
      until not (NamR[I-1] in [' '..'~'] ) or (I > 20);
      NamR[0] := Chr(I-1);
      if (Error = 0)
      then begin
              St:=dir + namR;
              AddElm(More, Courant, St);
      end;

    end;
    CutChain(Courant);
END;
(* - - - - - - - - -  fin de LectDir - - - - - - - - - - - - - - - - - - - - *)
```

MODULE GESTION ECRAN ET CLAVIER

```
procedure wrtps(horloge:ti; mo :integer );
(* Si HORLOGE contient des heures, minutes,secondes son contenu est
   affiche selon 3 MODES mo :
         m = 1:  Separes par des ":" => ex)  3:05:07
         m = 2:  avec des initiales  => ex)  3h 05m 07s
         m = 3:  avec des diminutufs => ex)  3hr 05min 07sec
         m = 4:  avec les noms       => ex)  3 Heures 05 Minutes 07 Secondes
*)
var  h, m, s : string[9];

BEGIN
     Case mo of
       1 : BEGIN
             h:=':' ;      m:=':' ;            s:= ' ' ;
           END;
       2 : BEGIN
             h:='h ' ;     m:= 'm ';           s:='s ';
           END;
       3 : BEGIN
             h:='hr ' ;    m:= 'min ' ;        s:= 'sec '
           END;
       4 : BEGIN
             h:= ' Heures ';
             m:= ' Minutes ';
             s:= ' Secondes ';
           END;
    END;

    write(horloge[1],h);
    IF (horloge[2] <= 9) THEN write('0');
    write(horloge[2], m);
    IF (horloge[3] <= 9) THEN write('0');
    write(horloge[3],s);
END;
(* - - - - - - - - - - - - fin de wrtps - - - - - - - - - - - - - - - - - *)


procedure TRACELIGNE(col, lgn, longueur : integer; car: char; verticale:boolean)
;

{ ARGT  : tous les parametres;
  ACTION: Trace une ligne de longueur LONGUEUR, composee du caractere CAR,
          A partir de la Colonne COL et la ligne LGN;
          Si VERTICALE est TRUE, la ligne sera verticale et horizontale Sinon.
}
var i,j : integer;

BEGIN
     IF (verticale) THEN
     BEGIN
          j:= lgn + longueur - 1;
          FOR i:= lgn TO j DO
          BEGIN
               gotoxy(col, i);     write(car);
          END
     END
     ELSE BEGIN
          j:= col + longueur - 1;
          gotoxy(col,lgn);
          FOR i := col TO j DO
          BEGIN
               write(car);
          END;
     END;
END;
```

```
(* -- - - - - - - - - - - - - fin de TRACELIGNE - - - - - - - - - - - - - - -*)
procedure NORMALV;

{ACTION : Permet que les prochaines ecritures a l'ecran se fassent caracteres
          blancs sur fond noir.
}
BEGIN
     textcolor(white); textbackground(black);
END;

(* - - - - - - - - - - - - - - fin de NORMALV - - - - - - - - - - - - - - *)

procedure INVERSV;

{ACTION : Permet que les prochaines ecritures a l'ecran se fassent caracteres
          noirs sur fond blanc.
}


BEGIN
     textcolor(black); textbackground(white);
END;

(* - - - - - - - - - - - - - - fin de INVERSV - - - - - - - - - - - - - - *)

procedure INVAFFICHE( col, lgn :integer; st: st80);

{ Affiche en mode inverse a partir de la colonne COL et de la ligne LGN
  le contenu de ST.
  EN SORTIE on est en mode normal a l'ecran.
}

BEGIN
     inversv;
     gotoxy(col, lgn);     write(st);
     normalv;
END;
(* - - - - - - - - - - - - - - fin de AFFICHE - - - - - - - - - - - - - - - - - *)

 procedure CENTRAGE(lgn, colmil: integer; st: stPhr );

{ ACTION : Centre le contenu de ST sur la colonne COLMIL a la ligne LGN.}

var   col : integer;

BEGIN
     col:= colmil - (length(st) div 2);
     gotoxy(col, lgn);
     write(st);
END;
(* - - - - - - - - - - - - - - fin de CENTRAGE - - - - -  - - - -  - -- - *)

procedure CADRE(cg, cd, lh, lb : integer);

{ PRE-COND : CG <= CD;
             LH <= LB;
             CG indice de colone Gauche;
             CD indice de colone Droite;
             LG indice de ligne haute;
             LB indice de ligne basse;
  ACTION    : TRACE un double cadre  passant par les points:
             (CG,LH); (CG,LB) ; (CD,LH); (CD,LB)
}
```

```
BEGIN
     gotoxy(cg,lh); write('à');
     gotoxy(cg,lb); write('ö');
     gotoxy(cd,lh); write('è');
     gotoxy(cd,lb); write('ü');
     traceligne(cg+1, lh, cd-cg-1, '¥', false);
     traceligne(cg+1, lb, cd-cg-1, '¥', false);
     traceligne(cg, lh+1, lb-lh-1, '£', true);
     traceligne(cd, lh+1, lb-lh-1, '£', true);
END;
(* - - - - -  - - - - - - - - fin de CADRE - - - - - - - - - - - - - - - - - *)

procedure EFFACECURSEUR;
{ PRE-COND : Le FOND est NOIR a lendroit du curseur;
  POSt-COND: La couleur des caracteres est noire.
  ACTION   : efface le curseur de l'ecran.
}

 var x, y : integer;

BEGIN
     x:= wherex; y:= wherey;
     textcolor(black); write(' ');
     gotoxy(x,y);
END;

(* - - - - - - - - - - - - - - fin de EFFACECURSEUR - - - - - - - - - - -  *)

function ENCORE : boolean;

{ ACTION : Attend QUE l'utilisateur tape ESPACE ou ESC;
           ENCORE = TRUE si on tape ESPACE
           ENCORE = FALSE si on tape ESC;
}


var   c : char;
   cont : boolean;

BEGIN
     cont:= true;
     effacecurseur;
     WHILE (cont) DO
     BEGIN
          reset(KBD); read(KBD,c);
          IF (not keypressed)
          THEN IF (c= ' ') OR ( ord(c) = 27)
               THEN BEGIN
                    textcolor(white); cont:= false;
                    encore:= (c= ' ');
               END;
     END;
END;
(* - - - - -  - - - -  fin de  ENCORE - - - - - - - - - - - - - - - *)

procedure EncoreEnter (var nobreak : boolean; var enter: integer);

{N'accepte de l'utilisateur que ESPACE, ENTER, ESC;
 On sort  des qu'il a tape ESC OU ENTER.
 ENTER = nbre de fois qu'il a tape enter avant de tapez ESC ou ESPACE.
 Si NOBREAK = TRUE, Il a fini par taper ESPACE
 SINON il a finit par taper ESC.
}


var   c : char;
```

```
   cont: boolean;

BEGIN
     enter:= 0;
     cont:= true;
     effacecurseur;
     WHILE (cont) DO
     BEGIN
          reset(KBD); read(KBD,c);
          IF (not keypressed)
          THEN IF (c= ' ') OR ( ord(c) = 27) { Blanc ou ESC. }
               THEN BEGIN
                    textcolor(white);
                    cont:= false;
                    NoBreak := (c=' ');

               END
               ELSE IF (ORD(c) = 13)  { ENTER }
                    THEN Enter:= Enter+1;
     END;


END;
(* - - - - -  - - - -  fin de  EncoreEnter - - - - - - - - - - - - - - - *)

procedure COIN;

{ ACTION : Place le curseur de l'ecran en (1,1);
}

BEGIN
     gotoxy(1,1);
END;

(* - - - - - -   - - - in de COIN - - - - -  -- - - - - - - -  *)

procedure ECRANBAS(stclig, st : st80; col, lgn : integer);

{Affiche le contenu de ST a la colonne COL et la ligne LGN.
 Le contenu DE STclig affiche a la colonne COL et la ligne LGN clignote.
}

VAR i: integer;

BEGIN
     inversv;  gotoxy(col,lgn); write(st);
     FOR i:= (length(st) + 1 ) TO (maxcol-4) DO write(' ');
     IF ((col + lgn)<> 0) THEN
     BEGIN
          textcolor( black + blink);
          gotoxy(col, lgn); write(stclig);
          coin;
     END;
     normalv;
END;
(* - - - - - - - - - - - fin de ECRANBAS - - - - - - - - - - - - - - *)

Procedure AffTab(tab : TabMot);

{ Affiche correctement @ l'{cran , le tableau TAB.}

VAR lgn , j, i : integer;
         Blanc : stMot;
BEGIN
     blanc:= '                                   ';
     i:= 0;      { Increment du NC de ligne. }
```

```
     j:= 0;      { Nombre de ligne d'ecran copiee.}

     WHILE (j < 8) DO
     BEGIN
          lgn:= 6+i;
          j:=j+1;

          centrage( lgn, 21, blanc);
          centrage( lgn, 58, blanc);

          centrage( lgn, 21, tab[j]);
          centrage( lgn, 58, tab[j+8]);
          i:=i+2;  { Car double interligne.}
     END;
END;
(* - - - - - - - - - - - - fin de AffTab - - - - - - - - - - - - - - - - - - - *)
procedure Window(l,c, nl,nc : integer);

{ A partir de la ligne L, et de la colonne C on cree une zone blanche
  de NL lignes et de NC colonnes.  Cette zone sera encadree par la procedure
  CADRE
}

Var   blk : stphr;
      i,j : integer;

BEGIN
     allblanc(blk);
     j:=nc+1;
     delete(blk,j,80);
     j:= l+nl;
     for i := 1 TO j DO
     BEGIN
          gotoxy(c,i);
          write(blk);
     END;
     i:=nc +c;
     j:=nl+1;
     cadre(c,i,l,j);
END;
{  - - - - - - - - - - - - - fin de WINDOW - - - - - - - - - - - - - - - -  -}

procedure TItre(col, lgn : integer; st : stPhr);

{ Efface l'ecran.
  Le contenu du String ST est centre sur la colonne COL, a la ligne LGN,
  en mode inverse, et est encadre.
  Les lignes du cadre sont separees de ST par 1 'Blanc'.
  EN SORTIE, l'ecran est en MODE NORMAL.
}
VAR   cg, cd, lh, lb : integer;

BEGIN
     clrscr;
     inversv;
     centrage(lgn, col, st);
     cg:= col-(length(st) DIV 2)-2;
     cd:=cg +length(st)+3;
     lh:=lgn -2;
     lb:= lgn +2;
     cadre(cg,cd,lh,lb);
     normalv;
END;
{ - --- - - - - - - - - - - fin de TITRE - - - - - - - - - - - - - - - - - }

Procedure saveScreen(VAR bc : PtBckUp; Var x, y : integer);
```

```
{ ACTION : Sauve l'ecran dans l'element pointe par BC,
           X et Y sont les indices de colonne et lignedu curseur
           avant le sauvetage;
}
BEGIN
     x:= wherex;
     y:= wherey;
     GetMem(bc,SizeScreen);
     move(DebEcran, bc^.Ecran, sizeScreen);
END;
(* - - - - - - - - - - - - - - Fin de sAVEsCREEN - -                    x)


Procedure CopyScreen(bc : PtBckUp; x,y : integer);

{ ACTION : copie a l'ecran le contenu de l'element pointe par bc.
           place le curseur en (x,y);
}

BEGIN
     move( bc^.Ecran, DebEcran, sizeScreen);
     FreeMem(bc,SizeScreen);
     gotoxy(x,y);

END;
(* - - - - - - - - - - - - - - Fin de CopysCREEN - - - - - -  - - - - - - *)

procedure fleche(haut, bas, col :integer);

{Affiche a la colonne COL, une fleche de la ligne HAUT a la ligne BAS.}

var i : integer;


begin
     gotoxy(col,haut);
     write('à¥¥¥¥)');
     for i:= haut+1 to bas-1 do
     begin
          gotoxy(COL,i);
          write(char(179));
     end;
     gotoxy(col - 5,bas);
     write('(¥¥¥¥');
     write(char(217));
end;
{---------------------------- fin de fleche ---------------------------- }

Procedure LireReponse (SetOfAnswer: SOA ; VAR Answer : integer);

{ ARGTS : SetOfAnswer
  Reslt : Answer;
  ACTION : SI SetOfAnswer contient la liste des reponse acceptable,
           alors answer est la reponse donnee.
}
VAR   cont : boolean;
   cAnswer : Char   ;

BEGIN
     cont:= true;
     WHILE (cont) DO
     BEGIN
          {reset(KBD);}
          Read(KBD, cAnswer);
```

```
          IF ( NOT KeyPressed )
          THEN BEGIN
              { Mettre en masjucules  .}
              IF (cAnswer >= 'a') AND (cAnswer <= 'z')
              THEN cAnswer:= chr(ord(cAnswer) - 32);

              IF ( (Ord(cAnswer) IN SetOfAnswer ) OR (Ord(cAnswer)= 27) )
              THEN Cont:= False;
          END;
     END;
     answer:= ord(cAnswer);
END;
(* - - - - - - - - - - fin de LireReponse - - - - - - - - - - - - - - - - *)

Procedure ACCEPT (VAR val : integer);

TYPE      StOfInt = Set OF 13..81;

CONST   Double : StOfInt = [72,73,75,77,80,81];
                                  { Valeur ASCII du Second caractere de
                                    respectivement:  Fleche vers le haut,
                                                     PG-UP, <-, ->,
                                                     Fleche vers le bas,
                                                     PG-ON
                                  }
        Simple : STOfInt = [13,27];
                                  { Valeur ASCII du 1er caractere de
                                    respectivement: RETURN, ESC.
                                  }



VAR         c1, c2 : Char    ;
            good : boolean   ;

BEGIN
     good:= False;
     REPEAT
           reset(KBD);
           read(KBD,c1);
           IF (KeyPressed)
           THEN BEGIN
                read(KBD,c2);
                val:= ord(C2);
                good:= ( val IN  double);
                val:= 27+val;
            END
            ELSE BEGIN
                 val:= ord(c1);
                 good:= (val IN Simple);
            END;
     UNTIL (good);

END;
(* - - - - - - - - - fin de Accept - - - - - - - - - - - - - - - - - - - *)
```

MODULE ACCES AUX RESULTATS

```
Procedure Initresult (nom : st24; io : integer );
{
ARGT    : Nom
Prcond : NOM contient le nom du fichier de resultats du lecteur.
         Le fichier de r{sultats est complet.
ACTION : IO est le code retour obtenu lors de l'utilisation du fichier de
         resultats.
         Si io = 0, recopie le contenu du fichier r{sultats dans les
         variables ad{quates.
}


VAR  f : file of integer;
   j,i : integer;
     r : result;
     D : PtInt;
BEGIN
{$i-}
     assign(f,nom);
     reset(f);
     io:= ioresult;
     IF (io = 0) THEN
     BEGIN
          seek(f,1);
          io:=ioresult;
     END;

     { INITIALISER LES VARIABLES SIMPLES}
     i:= 1;
     WHILE (io = 0) AND (i <= MaxSimple) DO
     BEGIN
         read(f,vsimple[i]);
         io:=ioresult;
         i:=i+1;
     END;

     {INITIALISER LES DATES }
     i:=1;
     WHILE (io = 0) AND (i <= MaxDate) DO
     BEGIN
         read(f,r);
         io:=ioresult;
         RealToDate(r,vDate[i]);
         i:=i+1;
     END;

     FOR i:= 1 TO MaxListe DO NilPtI(vListe[i]);

     {INITIALISER LA LISTE DE TMF }
     IF (io = 0) AND (1 <= Vsimple[2]) THEN
     BEGIN
         read(f,r);
         io:=ioresult;
         AjElmI(vListe[1], r);
     END;
     i:=1;
     D:= vListe[1];
     WHILE (io = 0) AND (i < Vsimple[2]) DO
     BEGIN
         read(f,r);
         io:=ioresult;
         AjElmI(d,r);
         i:=i+1;
     END;
```

```
{INITIALISER LES LISTES POUR LA LECTURE EN COLONNE }

FOR j:= 2 TO 4 DO
BEGIN
     IF (io = 0) AND (1 <= Vsimple[3]) THEN
     BEGIN
          read(f,r);
          io:=ioresult;
          AjElmI(vListe[j], r);
     END;
     i:=1;
     D:= vListe[j];
     WHILE (io = 0) AND (i < Vsimple[3]) DO
     BEGIN
          read(f,r);
          io:=ioresult;
          AjElmI(d,r);
          i:=i+1;
     END;
END;

{INITIALISER LES LISTES POUR LA LECTURE LA LECTURE DE TEXTE EN Y }

FOR j:= 5 TO 7 DO
BEGIN
     IF (io = 0) AND (1 <= Vsimple[4]) THEN
     BEGIN
          read(f,r);
          io:=ioresult;
          AjElmI(vListe[j], r);
     END;
     i:=1;
     D:= vListe[j];
     WHILE (io = 0) AND (i < Vsimple[4]) DO
     BEGIN
          read(f,r);
          io:=ioresult;
          AjElmI(d,r);
          i:=i+1;
     END;
END;

{INITIALISER LR POUR LA RECONNAISSANCE DE MOTS AU CONTOUR }

IF (io = 0) AND (1 <= Vsimple[5]) THEN
BEGIN
     read(f,r);
     io:=ioresult;
     AjElmI(vListe[8], r);
END;
i:=1;
D:= vListe[8];
WHILE (io = 0) AND (i < Vsimple[5]) DO
BEGIN
          read(f,r);
          io:=ioresult;
          AjElmI(d,r);
          i:=i+1;
END;

{INITIALISER LES LISTES POUR LA CONFUSION DE MOTS }

FOR j:= 9 TO 10 DO
BEGIN
     IF (io = 0) AND (1 <= Vsimple[6]) THEN
     BEGIN
```

```
                read(f,r);
                io:=ioresult;
                AjElmI(vListe[j], r);
           END;
           i:=1;
           D:= vListe[j];
           WHILE (io = 0) AND (i < Vsimple[6]) DO
           BEGIN
                read(f,r);
                io:=ioresult;
                AjElmI(d,r);
                i:=i+1;
           END;
     END;
{$i+}

END;
{ - - - - - - - - - - - - fin de InitResult - - - - - - - - - - }

Function GNDS : Integer;
{
 PreCond : InitResult a {t{ ex{cut{ avec succ}s.
 Action  : Rend le NUMERO DE LA DERNIERE SESSION.
}
BEGIN
     GNDS:=ROUND(vSimple[1]);
END;
{- - - - - - - - - - - - - - - - - Fin de GNDS - - - - - - - - - - - - - - - - - }


Function GNETMF : Integer;
{
 PreCond : InitResult a {t{ ex{cut{ avec succ}s.
 Action  : Rend le Nombre d'Elements dans la liste des TMF.
}
BEGIN
     GNETMF:=ROUND(vSimple[2]);
END;
{- - - - - - - - - - - - - - - - - Fin de GNETMF - - - - - - - - - - - - - - - - - }


Function GNELLC : Integer;
{
 PreCond : InitResult a {t{ ex{cut{ avec succ}s.
 Action  : Rend le Nombre d'Elements dans les Listes correspondant @ la
           Lecture en Colonne.
}
BEGIN
     GNELLC:=ROUND(vSimple[3]);
END;
{- - - - - - - - - - - - - - - - - Fin de GNELLC - - - - - - - - - - - - - - - - - }
Function GNELLPF : Integer;
{
 PreCond : InitResult a {t{ ex{cut{ avec succ}s.
 Action  : Rend le Nombre d'Elements dans les Listes correspondant @ la
           Lecture en par Points de Fixation.
}
BEGIN
     GNELLPF:=ROUND(vSimple[4]);
END;
{- - - - - - - - - - - - - - - - - Fin de GNELLPF - - - - - - - - - - - - - - - - - }
Function GNELRMC : Integer;
{
 PreCond : InitResult a {t{ ex{cut{ avec succ}s.
 Action  : Rend le Nombre d'Elements dans la Liste LR correspondant @ la
           Reconnaissance de Mots au Contour.
```

```
}
BEGIN
     GNELRMC:=ROUND(vSimple[5]);
END;
{- - - - - - - - - - - - - - - - - Fin de GNELRMC - - - - - - - - - - - - - - - - - }
Function GNELCM : Integer;
{
 PreCond : InitResult a {t{ ex{cut{ avec succ}s.
 Action  : Rend le Nombre d'Elements dans les Listes correspondant @ la
           Confusion de Mots .
}
BEGIN
     GNELCM:=ROUND(vSimple[6]);
END;
{- - - - - - - - - - - - - - - - - Fin de GNELCM - - - - - - - - - - - - - - - - - }
Function GMTMF : Integer;
{
 PreCond : InitResult a {t{ ex{cut{ avec succ}s.
 Action  : Rend la Moyenne des TMF de la liste.
}
BEGIN
     GMTMF:=ROUND(vSimple[7]);
END;
{- - - - - - - - - - - - - - - - - Fin de GMTMF - - - - - - - - - - - - - - - - - }
Function GMX : Integer;
{
 PreCond : InitResult a {t{ ex{cut{ avec succ}s.
 Action  : Rend le MX de l'{quation d'acceptation.
}
BEGIN
     GMX:=ROUND(vSimple[8]);
END;
{- - - - - - - - - - - - - - - - - Fin de GMX - - - - - - - - - - - - - - - - - }
Function GMN : Integer;
{
 PreCond : InitResult a {t{ ex{cut{ avec succ}s.
 Action  : Rend le MX de l'{quation d'acceptation.
}
BEGIN
     GMN:=ROUND(vSimple[9]);
END;
{- - - - - - - - - - - - - - - - - Fin de GMN - - - - - - - - - - - - - - - - - }
Function GCF : Integer;
{
 PreCond : InitResult a {t{ ex{cut{ avec succ}s.
 Action  : Rend le CF COURANT du lecteur.
}
BEGIN
     GCF:=ROUND(vSimple[10]);
END;
{- - - - - - - - - - - - - - - - - Fin de GCF - - - - - - - - - - - - - - - - - }
Function GPPCF : Integer;
{
 PreCond : InitResult a {t{ ex{cut{ avec succ}s.
 Action  : Rend le Plus Petit CF connu du lecteur.
}
BEGIN
     GPPCF:=ROUND(vSimple[11]);
END;
{- - - - - - - - - - - - - - - - - Fin de GPPCF - - - - - - - - - - - - - - - - - }
Function GPGCF : Integer;
{
 PreCond : InitResult a {t{ ex{cut{ avec succ}s.
 Action  : Rend le Plus Grand CF connu du lecteur.
}
BEGIN
```

```
     GPGCF:=ROUND(vSimple[12]);
END;
{- - - - - - - - - - - - - - - - Fin de GPGCF - - - - - - - - - - - - - - - - }
Function GMMLLLC : Integer;
{
 PreCond : InitResult a {t{ ex{cut{ avec succ}s.
 Action  : Rend la Meilleure Moyenne de LL pour la Lecture en Colonne.
}
BEGIN
     GMMLLLC:=ROUND(vSimple[13]);
END;
{- - - - - - - - - - - - - - - - Fin de GMMLLLC - - - - - - - - - - - - - - - - }
Function GMMLLLPF : Integer;
{
 PreCond : InitResult a {t{ ex{cut{ avec succ}s.
 Action  : Rend la Meilleure Moyenne de LL pour la Lecture par Points de
           Fixation.
}
BEGIN
     GMMLLLPF:=ROUND(vSimple[14]);
END;
{- - - - - - - - - - - - - - - - Fin de GMMLLLPF - - - - - - - - - - - - - - - - }
Function GPPTRMC : Integer;
{
 PreCond : InitResult a {t{ ex{cut{ avec succ}s.
 Action  : Rend le Plus Petit Temps enregistr{ pour la Reconnaissance de Mots
           au Contour.
}
BEGIN
     GPPTRMC:=ROUND(vSimple[15]);
END;
{- - - - - - - - - - - - - - - - Fin de GPPTRMC - - - - - - - - - - - - - - - - }
Function GPGTRMC : Integer;
{
 PreCond : InitResult a {t{ ex{cut{ avec succ}s.
 Action  : Rend le Plus Grand Temps enregistr{ pour la Reconnaissance de Mots
           au Contour.
}
BEGIN
     GPGTRMC:=ROUND(vSimple[16]);
END;
{- - - - - - - - - - - - - - - - Fin de GPGTRMC - - - - - - - - - - - - - - - - }
Function GMTCM : Integer;
{
 PreCond : InitResult a {t{ ex{cut{ avec succ}s.
 Action  : Rend le Meilleur Temps enregistr{ pour la Confusion de Mots.
}
BEGIN
     GMTCM:=ROUND(vSimple[17]);
END;
{- - - - - - - - - - - - - - - - Fin de GMTCM - - - - - - - - - - - - - - - - }
Function GNRMTCM : Integer;
{
 PreCond : InitResult a {t{ ex{cut{ avec succ}s.
 Action  : Rend le Nombre de Regressions pour le Meilleur Temps enregistr{
           pour la Confusion de Mots.
}
BEGIN
     GNRMTCM:=ROUND(vSimple[18]);
END;
{- - - - - - - - - - - - - - - - Fin de GNRMTCM - - - - - - - - - - - - - - - - }
```

MODULE DIAGNOSTIC

```
procedure TEMPS(var t:TI);

{ T contient l'heure : heure minutes secondes.}

VAR i, j, c :integer;
       tab : st6;

function time : st6;

TYPE  regpack = record
                ax, bx, cx, dx, bp, di, si, ds, es, flags:integer;
                end;

VAR  recpack              : regpack;
     ah, al, ch, cl, dh   : byte;
     hour, min, sec       : string[2];
BEGIN

     ah:=$2c;
     with recpack do
     BEGIN
          ax:=ah shl 8 + al
     END;
     intr($21,recpack);

     WITH recpack DO
     BEGIN
          str(cx shr 8, hour);
          str(cx mod 256, min);
          str(dx shr 8, sec);
     END;

     If length(hour) = 1 THEN hour:='0'+hour;
     If  length(min) =1 THEN min:='0'+min;
     If  length(sec)= 1 THEN sec:='0'+sec;
     time:=hour+min+sec;
 END;
       (* fin de la fonction TIMES *)

BEGIN
     tab:=time;
     i:=1; j:= 1;
     WHILE (i<= 3) DO
     BEGIN
          val(copy(tab,j,2), t[i],c);
          i:= i+1;  j:= j+2;
     END;
END;
(* - - - -  -- - - - - - - - fin de TEMPS - - - - - - - - - - - - - - - *)

procedure DATE(var t:TI);

{ En sortie : T contient la date : Jour, mois annee. }

VAR i, j, c :integer;
       tab : st8;

function day : st8;

TYPE  regpack = record
                ax, bx, cx, dx, bp, si, ds, es, flags:integer;
                end;

VAR  recpack              : regpack;
     month, jour          : string[2];
          year            : string[4];
```

```
          dx, cx       : integer;
BEGIN
     WITH recpack DO
     BEGIN
          ax:= $2a shl 8;
     END;
     MsDos(recpack);
     WITH recpack DO
     BEGIN
          str(cx, year );
          str(dx mod 256, jour);
          str(dx shr 8, month);
     END;
     IF length(jour)= 1 THEN jour:='0'+jour;
     IF length(month)=1 THEN month:='0'+month;
     day:= jour + month +year;
 END;
       (* fin de la fonction DAY *)

BEGIN
     tab:=day;
     i:=1; j:= 1;
     WHILE (i<= 2) DO
     BEGIN
          val(copy(tab,j,2), t[i],c);
          i:= i+1;  j:= j+2;
     END;
     val(copy(tab,5,4),t[3],c);
END;
(* - - - -  -- - - - - - - - fin de DATE - - - - - - - - - - - - - - - - *)

Function bissextil(i : integer): boolean;
{Si i est une annee bisssextile, BISSEXTIL = vrai sinon = faux}
BEGIN
     Bissextil:=(((I MOD 4)=0) AND ((I MOD 100) <> 0))  OR
                    ((I MOD 400) = 0);
END;
{ -------------------- fin de Bissextil ------------------------------}



Procedure DigitSeconde ( horloge:TI; var resultat : integer);

(*   Si HORLOGE contient l'heure, les minutes, les secondes , RESULTAT contient
         la valeur equivalente en secondes.  Les minutes et les secondes
         peuvent avoir des valeurs négatives.
     Si RESULTAT = Maxint, cela signifie que le calcul n'est pas possible
         sans provoquer d'overflow.
*)

(* Pour ne pas avoir d'overflow dans le calcul de nombre de secondes,
   la valeur maximale de horloge est 9h 05min 59sec.
   Si les minutes sont négatives, il est logique que le nombre d'heure
   puisse augmenter;  On accepte jusque 10 heures mais avec un maximum
   de  -55 minutes; donc 10h -55min 59 sec.
*)

BEGIN
     IF (horloge[1]>= 0) and (horloge[1]<= 8) or
     ((horloge[1] = 9 ) and (horloge[2] <= 5)) or
     ((horloge[1] = 10) and (horloge[2] <= -55))
     THEN
     resultat:= horloge[1] * 3600 + horloge[2] * 60 + horloge[3]
     ELSE
     resultat:= MAXINT;
```

```
     IF (resultat < 0) THEN resultat:= MAXINT;
END;
(*  - - - - - - - - - - - - - - fin de DigitSeconde - - - - - - - - - - - - - - - *)

Procedure SecondeDigit(secondes : integer; var horloge : TI);

(* Transforme le nombre SECONDES de secondes en son equivalent en heures
   minutes, et secondes.  (ex:3725 sec = 1h 02min 05sec)
*)

BEGIN
     horloge[3]:= secondes MOD 60;
       secondes:= secondes DIV 60;
     horloge[2]:= secondes MOD 60;
     horloge[1]:= secondes DIV 60;
END;
(* - - - - - - - - - - - - fin de SecondeDigit - - - - - - - - - - - - - - - - - *)


procedure SOUSTEMPS(t1, t2: TI; Var tr : TI; Var secondes : integer);
 (* Si T1, et T2 sont des heures,minutes, secondes en digital (ex 3:08:56)
    et si T2 est plus grand que t1, alors TR est le r‚sultat de(T2 - T1).
    Si T2 est 0:00:00, on consid‚rera T2 comme plus grand que T1 si T1<>T2.
    SECONDES est l'equivalent en secondes de TR.
 *)

VAR   change, cont : boolean;
                 i : integer;

BEGIN
     change:= false;  cont:= true;

     (* D'abord tenir compte d'un T2 = 00:00:00 *)

     IF (t2[1]= 0) AND (t2[2] = 0)  AND (t2[3] = 0)
     THEN IF (t1[1] = 0) AND (t1[2] = 0) AND (t1[3] = 0)
          THEN BEGIN
               tr[3]:= 0; tr[2]:= 0; tr[1]:= 0;
               secondes := 0;
               cont:= false;
          END
          ELSE BEGIN
               change:= true;
               t2[1] := 23; t2[2] := 59;  t2[3] := 59;
          END;
     IF (cont) THEN
     BEGIN
          FOR i:= 1 TO 3 DO  tr[i]:= t2[i]-t1[i];
          DigitSeconde(tr, secondes);
          IF (secondes <> MAXINT) THEN
          BEGIN
               IF (change) THEN secondes:= secondes + 1;
               SecondeDigit(secondes, tr);
          END;
      END;
END;
(* - - - - - - - - - fin de SOUSTEMPS - - - - - - - - - - - - - - - - - - - - - *)

procedure DiffTemps (var date1, date2, horlog1, horlog2, result: ti;
                     var secondes: integer);

(* Calcule la diff‚rence de temps entre l'heure HORLOG1 du DATE1, et
    l'heure HORLOG2 du DATE2;
     Le resultat en seconds se trouvent dans SECONDES, le r‚sultat
     digital en heures, minutes, et secondes se trouvent dans RESULT.
```

```
    La difference maximale permise est de 9h 05min 59sec.
    Dans le cas contraire, SECONDES prend la valeur maxint.
*)
TYPE  mois = set of 1.. 12;

VAR     a,b : real;
          i : integer;
       cm, lm: mois;
       minuit: ti;
         cont: boolean;

BEGIN
     cont:= true;

      (* Si on a change d'annee. *)

     IF (date2[3] - date1[3]) > 1 THEN cont:= false;

     IF (date1[3] <> date2[3]) AND cont
     THEN IF ((date1[1] <> 31) AND (date2[1] <> 1)) or
             ((date1[2] <> 12) AND (date2[2] <> 1))
          THEN cont:= false    else

      (* Si seul le mois a changé *)

     ELSE IF (date2[2] <> date1[2])
          THEN BEGIN
               lm:=[1,3,5,7,8,10,12];
               cm:=[4,6,9,11];
               IF (date1[2] IN lm)
               THEN IF ((date1[1] <> 31) or (date2[1] <> 1)) THEN cont:= false;
               IF (date1[2] IN cm)
               THEN IF ((date1[1] <> 30) or (date2[1] <> 1)) THEN cont:= false;
               IF (date1[2] = 2)
               THEN IF ((date1[1] <> 28) or (date2[1] <> 1)) THEN cont:= false;
          END

          (* Si seul le jour a change *)

          ELSE IF ((date2[1] - date1[1]) > 1) THEN  cont:= false;

     IF(not cont) THEN secondes:= MAXINT;

     IF (cont) THEN
     BEGIN

          IF (date1[1] = date2[1])
          THEN soustemps (horlog1, horlog2, result,secondes)
          ELSE BEGIN
               (* Calcul de horlog1 à minuit, puis de minuit à horlog2 *)

               for i:= 1 to 3 DO minuit[i]:= 0;
               soustemps(horlog1, minuit, result, secondes);
               IF (secondes <> MAXINT)
               THEN BEGIN
                    soustemps(minuit,horlog2, result, i);
                    IF (i = MAXINT)
                    THEN secondes:= MAXINT
                    ELSE BEGIN
                         a:=secondes;  b:= i;
                         a:=a/10; a:=a + (b / 10);
                         a:= a * 10;
                         IF (a > MAXINT)
                         THEN secondes := MAXINT
                         ELSE BEGIN
                              secondes:= secondes + i;
```

```
                              SecondeDigit(secondes,result);
                         END
                    END
               END
          END
     END
END;
(* - - - - - - - - - - - - - - - - in de DiffTemps - - - - - - - - - - - - - - *)

Procedure  BonTpsPfc(VAR jour1, Hor11, result1 : TI;
                     VAR duree1,codec, moitie, NbreLu: integer);

{ Pour Un excercice de lecture par POINT DE FIXATION :
SI (Nbrelu = MOITIE), alors on Calcule la difference de temps depuis
JOUR1 (jour,mois,annee), et  HORL1 (heure, min, sec).
Le resultat est mis dans RESULT et DUREE  (EN secondes).
Codec = 0 si TOUT va BIEN;
      = 1 Si Absurdement TROP PETIT;
      = 2 Si Absurdement TROP Grand;
      = 3 Si Trop petit pour etre dans intervalle acceptable;
      = 4 Si Trop Grand pour etre dans intervalle acceptable;
JOUR1 et HORL1 Represente le nouveau jour ,et la nouvelle heure.
}



var jour2, Hor12 : ti;


BEGIN
     codec:= 0;
     IF (NbreLu = Moitie) THEN
     BEGIN
          date(jour2);
          Temps(Hor12);
          DiffTemps(jour1, jour2, Hor11,Hor12, result1, duree1);
          { ici une procedure qui teste si la moyenne est acceptable.}
          date(jour1);
          Temps(Hor11);
     END;
END;
{ - - - - - - - - - - - - - - fin de   BonTpsPFc - - - - - - - - - - - - - - }

Procedure  BonTpsClc(VAR jour1, Hor11, result1 : TI;
                     VAR duree1,codec, moitie, NbreLu: integer);

{ Pour Un excercice de lecture en COLONNE :
SI (Nbrelu = MOITIE), alors on Calcule la difference de temps depuis
JOUR1 (jour,mois,annee), et  HORL1 (heure, min, sec).
Le resultat est mis dans RESULT et DUREE  (EN secondes).
Codec = 0 si TOUT va BIEN;
      = 1 Si Absurdement TROP PETIT;
      = 2 Si Absurdement TROP Grand;
      = 3 Si Trop petit pour etre dans intervalle acceptable;
      = 4 Si Trop Grand pour etre dans intervalle acceptable;
JOUR1 et HORL1 Represente le nouveau jour ,et la nouvelle heure.
}



var jour2, Hor12 : ti;


BEGIN
```

```
     codec:= 0;
     IF (NbreLu = Moitie) THEN
     BEGIN
          date(jour2);
          Temps(Hor12);
          DiffTemps(jour1, jour2, Hor11,Hor12, result1, duree1);
          { ici une procedure qui teste si la moyenne est acceptable.}
          date(jour1);
          Temps(Hor11);
     END;
END;
{ - - - - - - - - - - - - - - fin de   BonTpsClc - - - - - - - - - - - - - - - }

function nbrjour(mois1, an : integer): integer;

{ PRE-Cond : Mois1 est un numero de mois de l'annee AN.
  NbrJour est le nombre de jours du mois MOIS1.
}

var i : integer;
   m31,M30 : mois;

BEGIN
     M31:=[1,3,5,7,8,10,12];
     M30:=[4,6,9,11];

     IF(mois1) IN M31
     THEN i:= 31
     ELSE IF ((mois1) IN M30)
          THEN i:= 30
          ELSE IF(bissextil(an))
               THEN i:= 29
               ELSE i:= 28;
     NbrJour:= i;
END;

{ ------------------------ fin de NbrJour ------------------------- }

procedure DiffMois(mois1, mois2, an : integer;   var jour :real);

{Pre-Cond : MOIS1, et Mois2 sont des mois d'une meme annee.
            ANNEE contient le nombre de jours de chaque mois de l'annee.
            0<= Mois1 <=12 ET 1 <= MOIS2 <=13

 Post-condition : JOUR est incremente du nombre de jours qu'il y a entre
                  Le 1er du MOIS1 et Le dernier du MOIS2
}

BEGIN

     WHILE MOIS1 <= MOIS2 DO
     BEGIN
          jour:=nbrJour(mois1,an) + jour;
          mois1:=mois1+1;
     END;
END;
{------------------------------- fin de Diffmois -------------------- }
procedure DiffAn (an1, an2 : integer; var jour : real);

{ JOUR est incremente du nombre de jours qu'il y a entre le 1er de AN1,
       et le dernier AVANT AN2.
}
BEGIN
     WHILE (an1 < An2) DO
     BEGIN
          IF Bissextil(An1)
```

```
          THEN Jour:= jour+366
          ELSE jour:= jour+365 ;
          an1:=an1+1;
     END;
END;
{-------------------------------- fin de diffan ---------------------- }

procedure diffjour(d1,d2 : ti; var jour: real);
{ pre-cond : D1 et D2 sont des dates : jour, mois, annee
             D1 <= D2
  argt     : D1,D2
  reslt    : jour
  post-cond: Jour = le nombre de jours de difference entre D1 et D2
}

BEGIN
     IF (D1[3] <> D2[3])
     THEN BEGIN
          jour:= nbrJour(D1[2],D1[3]) - D1[1];
          IF (D1[2] < 12) THEN DiFFMois(D1[2]+1, 12, D1[3], jour);
          diffAn(D1[3]+1, D2[3], jour);
          diffMois(1,D2[2]-1, D2[3], jour);
          jour:= jour+ d2[1];
      END
      ELSE IF (D1[2] <> D2[2])
          THEN BEGIN
               jour:= nbrJour(D1[2],D1[3]) - D1[1];
               DiFFMois(D1[2]+1, D2[2]-1, D1[3], jour);
               jour:= jour+ d2[1];
          END
          ELSE jour:= D2[1]-D1[1];
END;
{------------------------ fin de DiffJour ------------------------------ }
Procedure RealToDate (Rdt : real; VAR dt : TI );
{
 Argt     : RDT
 reslt    : Dt
 Prcond   : RDT est la repr{sentation sous forme de r{el d'un date, grâce @ la
            formule RDT= (ANNEE * 10000) + (MOIS * 100) + JOUR.
PostCond : Dt est un tableau de trois entiers representant un jour, mois,annee
            obtenu @ partir de Rdt. L'annee est > 0.
}
Var r, re : real;

BEGIN
     r:= rdt /10000  ;
     dt[3]:= trunc(r);

     re:= dt[3];
     r:= re*10000;
     rdt:= rdt-r;
     r:= rdt/100;
     dt[2]:= trunc(r);

     re:= dt[2];
     r:= re*100;
     rdt:= rdt - r;
     dt[1]:= trunc(rdt);
END;
{ - - - - - - - - - - - - - - - fin de REALTODATE- - - - - - - - - - - - - - - - }
Procedure DateToReal (dt : ti; VAR Rdt : real );
{
 Argt     : Dt
 reslt    : Rdt
 Prcond   : Dt est un tableau de trois entiers representant un jour, mois,annee
            obtenu @ partir de Rdt. L'annee est > 0.
```

L.5

```
PostCond :RDT est la repr{sentation sous forme de r{el d'un date, gr}ce @ la
          formule RDT= (ANNEE * 10000) + (MOIS * 100) + JOUR.

}
Var r : real;

BEGIN
     r:=dt[3];
     rdt:= (r*10000) + (Dt[2] * 100) + Dt[1];
END;
{ - - - - - - - - - - - - - - - fin de DateToReal - - - - - - - - - - - - - - - - }
```

MODULE TRAITEMENT DE TEXTE

```
Procedure MkSt(Var S, ST : Stmot; VAR NbCar, qte : integer; VAR f : frec );

{PRECOND : F est assigne et ouvert.
 ARGT : S,Nbcar,f
 RESLT : S, ST, qte
 ACTION: Si S <> '#', alors ST sera la concatenation de S et de strings
         contenus dans le fichier F.
         La longueur totale de ST ne dépasse pas NBCAR caracteres.
         Si en sortie S <> '#', alors S contient le dernier mot lu dans F,
         mais pas concaténé car trop long.
}
VAR     r : rec;
      cont: boolean;

BEGIN
     IF (s <>'#') THEN
     BEGIN
          st:= s;
          qte:= qte+1;
          IF not eof(f)
          THEN BEGIN
               read(f, r);
               s:=r.elm;

          END
          ELSE s:= '#' ;

          cont:= true;
          WHILE (Cont) AND (s <> '#') DO
          BEGIN
               IF (NbCar >= (Length(s) + length(st) + 1))
               THEN BEGIN
                    St:= St +' '+s ;
                    qte:=qte+1;
                    IF NOT(EOF(f))
                    THEN BEGIN
                         read(f,r);
                         s:=r.elm;
                    END
                    ELSE s:= '#'
               END
               ELSE cont:= false;
          END;
     END;

END;

{- - - - -  - - - - - - - - -- fin de Mkst  - - - - - - - - --  - - - - - -- - - }
Procedure MakeY(var f,g : frec; CF : Integer);

{PRECOND : F et G sont fermes et assignes.
 ARGT : F,CF
 RESLT : G
 ACTION: Si F est un fichier traite par la procedure FILEMOT, alors le
         Fichier G contiendra la structure en "Y" du texte, pour un champs
         de fixation CF.
}

Var
                                        Sg, Sd, s : StMot;
                                        r         : rec;
    NbCar, i, pas,combien, qte,coupure, longueur  : integer;
                                          moitie : real;
    { Nbcar : Taille maximum d'une demi-ligne.
      Pas   : Pas de decrementation du nombre de blancs entre 2 demi-lignes.
```

```
       qte   : Nbre d'elements de F deja traites.
    combien  : Nbre de lignes de G deja traitees.
    coupure  : NI de la 1ere ligne qui Subit un accroissement de Nbcar
     moitie  : Nbre d'element de F, a partir duquel il convient d'incrementer
               NBCAR.
     }

     change  : boolean; { Indique si on a deja incremente NBCAR.}
        res  : stPHR;


Procedure MAJ (Var   pas, coupure : integer; var change : boolean;
                  combien, qte, nbcar: integer; moitie : real);

{ ARGT : Change, qte, moitie, combien, moitie
 RESLT : CHANGE, PAS, COUPURE
 ACTION:  Si on a pas encore incremente NBCAR (CHANGE = FALSE),
          alors MAJ met-a-jour PAS, CHANGE, COUPURE
}

BEGIN
          IF (NOT change)
          THEN BEGIN
               IF (QTE > Moitie)
               THEN BEGIN
                    NbCar:= NbCar + incr;
                    IF (nbcar > MxLgMot) THEN nbcar:= MxLgMot;
                    change:= true;
                    coupure:= combien;
               END;
               IF (((MxLgPhr - (2*NbCar)) DIV NbLn) > 1)
               THEN pas:= 3
               ELSE pas:= 2;

          END;
END;
{- - - - - - - - - -  fin de MAJ - - - - - - - - - - - - - - - - - - - - - - - }
Procedure INITmakeY;{ Effectue les initialisations utiles a MakeY}
BEGIN
     reset(f);
     rewrite(g);
     str(0,r.elm);
     write(g,r);
     moitie:= filesize(f) *( CF / (2*CF+incr));
     qte:= 0;
     combien:= 0;
     NbCar:= Cf;
     change:= false;

END;
{  ---------------------------- fin de InitMakeY  ---------------------- }

BEGIN
     initmakey;
     IF (not EOF(f))
     THEN BEGIN
          read(f,r);
          s:=r.elm;
     END
     ELSE s:= '#';

     WHILE (s <> '#') DO
     BEGIN
          MAJ (  pas, coupure ,change, combien, qte, nbcar,moitie );
          i:= 1;
```

```
          {Rapprochement des demis-phrases}
          WHILE (i <= NbLn) AND (s <> '#') DO
          BEGIN
               MkSt(s,sg,NbCar, qte,f);
               MkSt(s,sd,NbCar,qte,f);
               longueur:= MxLgPhr - (i* pas);
               MkLine(Sg,Sd,res,longueur);
               r.elm:=res;
               write(g,r);
               combien:= combien+1;
               i:=i+1;
           END;

          i:= 1;

          {Ecrire les demi-phrases reunies}
          WHILE (i <= NbLn) AND (s <> '#') DO
          BEGIN
               MAJ (  pas, coupure ,change, combien, qte, nbcar,moitie );
               MkSt(s,sg,NbCar, qte,f);
               MkSt(s,sd,NbCar,qte,f);
               r.elm:=sg+' '+sd;
               write(g,r);
               combien:=combien+1;
               i:= i+1;
          END;

     END;
     seek(g,0);
     str(coupure,r.elm);
     write(g,r);
     close(g);
     close(f);
END;
{ ------------------------ fin de MAKEY ------------------------------ }

Procedure buz;

begin
          sound(80); delay(500);
          nosound;
end;
(* - - - - - - - - - - - - fin de buz - - - - - - - - - - - - - - - - - *)

procedure ALEATOIRE( min, max : integer ;var result : integer);

{ Rend un nombre aleatoire positif compris entre MIN et MAX .}

VAR i : integer;

BEGIN
     i:= max - min + 1;
     i:= random(i);
     result:= i+min;
END;
{ - - - - - - - - - - fin de ALEATOIRE - - - - - - - - - - - - - - - - - }



Procedure FileMot (var t:text; var f: frec);

{ARGT : T
 Reslt: F
 Si T contient un texte, alors F contient la decomposition mot a mot de ce
```

L.S

```
  texte.)
Var
     entre, fin : stPhr;
           cont : boolean;
              r : rec;
BEGIN

     reset(t);
     rewrite(f);
     WHILE NOT EOF(t) DO
     BEGIN
          readln(t,entre);

          StversMots(entre,fin,cont);
          WHILE cont DO
          BEGIN
               r.elm:= fin;
               write(f,r);
               StVersMots(entre, fin,cont);
          END;
       END;

       close(f);
       close(t);
END;
{ ---------------------- fin de FileMot ----------------------------}
```

MODULE PRESENTATION

```
Procedure presentation(VAR T : text; Tit : StPhr);

{ PRE-COND : Le fichier T existe, et est assigne.
  ARGT     : T, Tit
  ACTION   : Affiche le contenu du fichier T a l'ecran, ligne par ligne, a la
             vitessede l'utilisateur.
             Au-dessus de l'ecran est affiche en permanence le contenu de TIT.
             Le titre sera change des qu'on rencontre dans T, un string dont la
premiere
             lettre est un '#'; Le reste de ce string sera ignore.
             Le nouveau titre sera le contenu du string suivant.
}

CONST
     PremLgn = 7;  { Indice de la 1ere ligne affichee.}
     DernLn = 23;  { Indice de la derniere ligne affichee.}
     ColMil = 40;  { Indice de la colonne centrale.}
     LgnTit = 3 ;  { Indice de la ligne du titre.}
     Last   = 25; {Indice de ligne de la legende.}
     np     = 12; {Indice de la ligne ou on affiche 'nouveau paragraphe'}
VAR
   i       : integer;
   legende, St : StPhr;
   cont, newPar  : boolean;


BEGIN
     clrscr;
     newPar := false;

     LEGENDE:=' B.E.: POURSUIVRE';
     Legende := Legende + '                                                  ';
     legende := legende + 'ESC : INTERROMPRE ';

     RESET(T);
     cont:= not(eof(T));
     IF cont THEN  readln(t,st);
     WHILE cont DO
     BEGIN
          { M-A-J du TITRE. }

          IF (POS('#',ST) = 1) THEN
          BEGIN
               cont:= NOT(eof(T));
               IF (cont) THEN
               BEGIN
                    readln(T,ST);
                    Tit:= st;
                    cont:= NOT(eof(t));
                    IF (cont) THEN Readln(T,St);
               END;
          END;

          { Affichage des titres et legendes.}

          IF (CONT) THEN
          BEGIN
               Titre(ColMil, LgnTit, Tit);
               InvAffiche(1,Last,legende);

               IF(newpar) THEN
               BEGIN
                    centrage(np,Colmil,'Nouveau PARAGRAPHE');
                    cont:= encore;
                    gotoxy(1,np);
```

```
                    ClrEol;
               END;
          END;

          {Affichage des lignes.}

          i:= PremLgn;

          WHILE (cont) AND (Pos('#',st) <> 1) DO
          BEGIN
               gotoxy(1,Last);     DelLine;
               gotoxy(1, PremLgn);  DelLine; DelLine;

               WHILE (cont)  AND (i <= DernLn) AND (Pos('#',st) <> 1) DO
               BEGIN
                    Gotoxy(1,i);
                    write(St);
                    cont:= not(eof(T));
                    IF (cont) THEN readLn(T,St);
                    i:=i+2;
               END;

               InvAffiche(1,Last,legende);

               IF NOT(cont) THEN
               BEGIN
                    centrage( maxlgn-1,colmil,'---- FIN DE TEXTE ----');
                    buz;
               END;

               cont:= cont and encore;
               i:= i-2;

          END;
          newpar:= true
     END;
     close(t);
END;

{--------------------    fin de presentation - - - - - - - - - - - - - - - - - - -
}

Procedure motamot(var f : frec; var t : text;
                  var duree, duree2  : integer; VAR result, result2 : TI;
                  VAR NoBreak : Boolean
                  );


{PRE-Cond : Chaque element de F, est un mot du texte T;
ARGT      : f,t
Reslt     : Duree, duree2,Result, result2, Nobreak
Post-Cond:
            NoBreak = True, s'il n'y a pas eut de demande d'interruption;
                    = False sinon.
             Result est du type : Heure, min,sec;
             Duree  est un nombre de SECONDES.
             RESULT,DUREE, et RESULT2, DUREE2 sont respectivement les resultats
             pour la lecture mot a mot , et la lecture naturelle.

}
CONST   prem = 4;   { Indice de 1ere ligne.}
        Last = 24;  { Indice de Derniere ligne.}
         leg  = 2;   { indice ligne de legende .}
      colmil = 40;    { Indice de colonne centrale.}
      coldr  = 56;  { indice de colonne pour seconde rangee de mot.}
VAR
```

```
                                        i : integer;
                                     cont : boolean;
                  jour1, jour2, hor11, hor12 : TI     ;
                              legende, s : StPhr  ;
                                        r : rec   ;


BEGIN
    Clrscr;

    duree:=0;
    duree2:=0;

    reset(f);
    reset(t);

    cont:= not(eof(f)) and not (eof(t));
    IF (cont) THEN
    BEGIN
         legende:=' B.E. : Declencher
ESC : Interrompre ';
         Invaffiche(1,leg,legende);
         legende:=' B.E. : Poursuivre
ESC : Interrompre ';
         nobreak:= encore;
         cont:=nobreak;
         IF (Cont) THEN
         BEGIN
              Invaffiche(1,leg,legende);

              { Prendre l'heure de depart.}
              date(jour1);
              temps(hor11);
         END;

         WHILE CONT DO
         BEGIN
              i:= prem;
              WHILE(cont) AND  (i <= last) DO
              BEGIN
                read(f,r);
                cont:= not(eof(f));
                gotoxy(1,i);
                clreol;
                writeln(r.elm);
                i:= i+2;
              END;

              IF (cont)
              THEN fleche( prem,last ,colmil)
              ELSE traceLigne(colmil,prem,last-prem,' ',true);
              WHILE (i <= last) DO
              BEGIN
                   gotoxy(1,i);
                   clreol;
                   i:=i+2;
              END;

              i:= prem;
              WHILE(cont) AND (i <= last) DO
              BEGIN
                read(f,r);
                cont:= not(eof(f));
                gotoxy(coldr,i);
                clreol;
                writeln(r.elm);
```

```
                i:= i+2;
            END;
            nobreak:= encore;
            cont:= cont AND NoBreak;
        END;

        { Eteindre le chrono.}
        temps(hor12);
        date(jour2);

        { Calculer le temps. }
        difftemps(jour1, jour2, hor11, hor12, result,duree);
        cont:= noBreak;
    END;


        { FIN de l'affichage par mot.}
        { Affichage naturel }

    IF (cont) THEN
    BEGIN
        clrscr;
        legende:=' B.E. : Declencher
ESC : Interrompre ';
        Invaffiche(1,leg,legende);
        legende:=' B.E. : Poursuivre
ESC : Interrompre ' ;
        nobreak:= encore;
        cont:=nobreak;

        IF (Cont) THEN
        BEGIN
            Invaffiche(1,leg,legende);

            { Prendre l'heure de depart.}
            date(jour1);
            temps(hor11);
        END;

        WHILE CONT DO
        BEGIN
            i:= prem;
            WHILE(cont) AND  (i <= last) DO
            BEGIN
              readln(t,s);
              cont:= not(eof(t));
              gotoxy(1,i);
              clreol;
              writeln(s);
              i:= i+2;
            END;

            WHILE (i <= last) DO
            BEGIN
                gotoxy(1,i);
                clreol;
                i:=i+2;
            END;
            nobreak:= encore;
            cont:= cont AND NoBreak;
        END;

        { Eteindre le chrono.}
        temps(hor12);
        date(jour2);
```

```
        { Calculer le temps. }
        difftemps(jour1, jour2, horl1, horl2, result2,duree2);
        cont:= noBreak;
    END;
    close(f);
    close(t);
END;
(* - - - - - - - - - - - - fin de MotaMot - - - - - - - - - - - - - - - - - - - *)


Procedure AffMot(Depart : PtStMt;   delais : integer;
                 var duree, duree2  : integer;
                 VAR result, result2 : TI;  Demi : integer;
                 VAR NoBreak : Boolean; Var codec : integer
                  );


{PRE-Cond : DEPART pointe sur une chaine contenant 2 fois DEMI MOTS;
            Delais )= 0;
ARGT      : Depart, DELA, Demi, I
Reslt     : Duree, duree2,Result, result2, Nobreak, Codec
Post-Cond:
            NoBreak = True, s'il n'y a pas eut de demande d'interruption;
                    = False sinon.
            Codec   = 0, Si tout va bien pour les DEMI premiers mots.
                    = 1, Si Le tps est absurdement trop court pour les
                         DEMI premiers mots.
                    = 2  Si Le tps est absurdement trop long pour les
                         DEMI premiers mots.
                    = 3  Si Le tps est Sous l'intervalle acceptable pour les
                         DEMI premiers mots.
                    = 4
             RESULT,DUREE, et RESULT2, DUREE2 sont respectivement les resultats
             pour les DEMIS PREMIERS, (DERNIERS) MOTS .
Action    :  Si Delais = 0, les mots apparaissent apres que l'utilisateur
                tape sur la barre d'espacement.
             Sinon, chaque nouveau mot apparait apres DELAIS milisecondes.

}

VAR                                 mesg, mesclig : st80   ;
                                        nombre, i : integer;
                                        eol, cont : boolean;
              jour1, jour2, horl1, horl2, tab3 : TI      ;
                                             Mot : StMot   ;


PROCEDURE POSE( VAR jour1, Horl1, Result : TI;  Delais : integer;
                VAR Duree, Demi, QTE, codec : integer;
                VAR Nobreak, cont : boolean
              );

{
 ARGT      : Jour1, Horl1, Delais, demi, qte, cont
 result    : jour1, Horl1, Result, Duree, Codec, NoBreak, Cont
 Post-Condition :
                Cont = Cont And Nobreak And (Codec= 0).
                Quand DEMI = QTE :
                    RESULT et DUREE sont mis a jour;
                    Jour1 et Horl1 sont les nouvelles valeurs du chrono.

                Nobreak est a jour;
Action    : Si DELAIS = 0,  attend que le lecteur tape Espace ou ESC.
            Sinon attend DELAIS milisecondes avant d'afficher le prochain mot.
            Met a jour , result et duree, si demi = qte.
```

```
BEGIN
     IF (delais) = 0 THEN
     BEGIN
          NoBreak:=Encore;
          BonTpsClc(jour1, Horl1, result, duree,codec, demi, qte);
          Cont:= Cont  AND NoBreak AND (Codec = 0);
     END
     ELSE delay(delais);
END;

{------------------------ fin de POSE ------------------------------------}


{'ÿ' a le code 209 }

BEGIN

     Clrscr;
     cadre(2, 77, 2, 22);
     centrage(2, 21, 'ÿ');
     centrage(2, 58, 'ÿ');
     traceligne(40, 3, 19, '£', true);

     { Tracer les points au milieu des colonnes.      }
     i:= 4;
     WHILE (i<= 20) DO
     BEGIN
          centrage(i,21,'.');
          centrage(i,58,'.');
          i:=i+2;
     END;

     {Afficher le menu au bas de l'ecran, avec "Mot Suivant " qui clignote. }
     mesclig:='  B.E. : Mot suivant ';
     mesg:= mesclig + '                                          '+'  ESC : Interrompre';
     ecranbas(mesclig, mesg, 2, 24);
     nobreak:= encore;

     IF (NoBreak) THEN
     BEGIN
         nombre:= 0;

        { Arreter le Clignotement    }
        IF (delais > 0) THEN mesclig:= '                    ';
        invaffiche(2, 24, mesclig);

        { Prendre l'heure de depart.}
        date(jour1);
        temps(horl1);

        initlist(depart,eol);
        cont:= not(eol);

        WHILE CONT DO
        BEGIN
            i:= 3;
            WHILE(cont) AND (i <= 21) DO
            BEGIN
                 nextlect(depart,eol,mot);
                 cont:= not(eol);
                 centrage(i, 21, mot);
                 {
                 coin;
                 }
```

```
                i:= i+2;
                nombre:=nombre+1;
                POSE(  jour1, Hor11, Result , Delais , Duree, Demi, nombre, cod
e,Nobreak, cont);

          END;

          i:= 3;
          WHILE(cont) AND (i <= 21) DO
          BEGIN
                NextLect(depart,eol, mot);
                cont:=not(eol);
                centrage(i, 58, mot);
                {
                coin;
                }
                i:= i+2;
                nombre:= nombre+1;
                POSE(  jour1, Hor11, Result , Delais , Duree, Demi, nombre, cod
e,Nobreak, cont);

          END;

          { Eteindre le chrono.}
          temps(hor12);
          date(jour2);



          { Blanchir le reste des colonnes.}
          i:= 3;
          WHILE(i<= 21) AND (cont) DO
          BEGIN
                centrage(i,21,'                                        ');
                centrage(i,58,'                                        ');
                i:=i+2;
          END;
     END;


     { Calculer le temps. }
     difftemps(jour1, jour2, hor11, hor12, result2,duree2);

  END;
END;
(* - - - - - - - - - - - - fin de AffMot - - - - - - - - - - - - - - - - - - *)

Procedure Question (VAR cont : boolean;
                    VAR Bonnereponse, TotalQuest, io : integer;
                    VAR TotalSigne, TotalSeconde      : real   ;
                        Name                          : StMot
                   );

{ PRE-COND : NAME est le nom d'un fichier texte dont le contenu a la structure
             suivante :
             (debut texte)

             (fin texte)
             #abcd
             { texte de la premiere question, acceptant les reponses possibles:
               a,b,c,d; et 'a' est la bonne reponse.
             }
             (fin de fichier)

             Pour chaque nouvelle question, un cardinal suivit des reponses
             possible, avec comme bonne reponse le caractere suivant le #.
```

```
            Ce caractere ne peut pas etre blanc.
  POST-COND: IO = Le code d'entree sortie indiquant s'il y a eut un probleme
             avec le fichier dont le nom est dans NAME.
             SI CONT = FALSE, le lecteur a interrompu
             SINON TOTalsigne = le nombre de signes du texte.
                   TotalSeconde = le temps de lecture en secondes.
                   BonneReponse = le nombre de bonnes reponses.
                   TOtalQuest   = le nombre de questions.
}


CONST   MaxNrLigne = 23;


VAR     i,j, NrLigne, Seconde, Cardinale, Answer,
                                  Solution, Confirmation : integer;
        Correct, More, PasQuestion                       : boolean;
        StClig, Message, Ligne, WaitLigne
: StPhr  ;
        F                                                          : Text    ;
        Jour1, Jour2, Hor11, Hor12, Result                         : TI      ;
         SetOfAnswer                                               : SOA     ;



Procedure TrouvQuest (Var io: integer;  Var f : text ;  Var correct : boolean;
                      Var Ligne, WaitLigne : StPhr
                     );

{ Pre-Conditions : LINE contient un '#'.
                  F est un fichier ouvert dont le dernier element lu
                  est LINE.
  ARGTS          : f, ligne
  reslt          : io,correct, ligne, waitligne

  POST-Cond      : Si Correct = TRUE ET io = 0
                     LINE contient la ligne contenant UN cardinal suivit d'au
                     moins un caractere non-blanc. Cette ligne n'est pas suivie
                     d'une autre qui contient au moins un cardinal.
                     WAITLIGNE Contient la ligne suivant LINE dans F.
                   SINON Le Fichier F n'avait pas la bonne structure.
                   IO contient le code d'entree sortie pour le fichier F.
}

Var         i : integer;
        more  : boolean;

BEGIN
  Correct:= false;

  IF ( not(eof(f)) AND (io = 0) )
  THEN BEGIN
       i:= Pos('#',ligne) + 1;
       Correct:= ( (i<=length(ligne)) AND (ligne[i] <> chr(32)) );
       WaitLigne:= '';
       {$i-} readln(f,WaitLigne); {$i+}
       io:= ioresult;
       correct:= Correct AND (pos('#',WaitLigne) = 0);
       more:= true;

       WHILE (io = 0) AND (More) AND (not(correct)) DO
       BEGIN
            ligne:=WaitLigne;
            i:= pos('#', ligne) + 1;
            IF (i > 1)
```

```
            THEN correct:=( (Not(eof(f))) AND ( i<= length(ligne) )
                                          AND ( ligne[i] <> chr(32) )
                          )
            ELSE correct:= false;

            More:= not(eof(f));
            WaitLigne:= '';

            IF (more) THEN
            BEGIN
                 readln(f,WaitLigne);
                 io:= ioresult;
                 correct:= correct AND (pos('#',WaitLigne) = 0);
            END;
       END;
  END;
END;
{ - - - - - - - - - - - FIN de TrouvQuest - - - - - - - - - - - - - - - - - - - - - -}

Procedure TrouvSolution (ligne : StPhr; Var solution : integer;
                                        Var SetOfAnswer : SOA

                         );

{Pre-Con : Recoit LiGNE qui contient un '#' suivi d'au-moins un
           caract}re non-blanc.
 ARGT    : Ligne;
 Resultat: Solution, SetOfAnswer

POST-COND: Le caractere suivant le cardinal est mis dans SOLUTION.
           Tous les caracteres, suivant le cardinal, sans etre separe
           par un ou des blancs, se trouvent dans SetOfAnswer

}

VAR i, j : integer;

BEGIN
     i:= pos('#',ligne) + 1;
     j:= ord(ligne[i]);

     { Mise en majuscule. }
     IF ( (j >= Ord('a')) AND (j <= ord('z')) ) THEN j:= j- 32;

     Solution:= j;
     SetOfAnswer:=[];

     WHILE ( i <= length(ligne) ) AND (ligne[i] <> chr(32) ) DO
     BEGIN
          j:= ord(ligne[i]);

          IF ( (j >= Ord('a')) AND (j <= ord('z')) ) THEN j:= j- 32;

          SetOfAnswer:= SetOfAnswer + [j];
          i:=i+1;
     END;
END;

(* - - - - - - - - -  fin de TrouvSolution - - - - - - - - - - - - - - - - - *)

Procedure AffTexte( Var f : text; Var io : integer;  Var Ligne : StPhr );

{ Pre-condition : Ligne ne contient pas de '#',et est le dernier element
                  lu dans F.
  Argt          : f, ligne
  reslt         : f, io,ligne
  POST-Cond     : SI io = 0 alors
                       Si Not(eof(f)) ,Ligne est la premiere suivante de F
```

```
                                        contenant un #.
                          SINON Ligne contient la derniere ligne de F
  Action          : Affiche Ligne et au maximum les MaxNrLigne lignes suivantes
                    de F, tant qu'elles ne contiennent pas de Cardinale.
  }

VAR  Cardinal, NroLigne : integer;
     NoCardinal         :  boolean;

BEGIN
     clrscr;
     Writeln(ligne);
     NoCardinal:= TRUE;
     NroLigne:= 1;

     WHILE ( (not(eof(f))) AND
             (io = 0)      AND
             (NoCardinal)  AND
             (NroLigne < MaxNrLigne) ) DO
     BEGIN
             NroLigne:= Nroligne+1;
             {$i-}  readln(f,ligne); {$i+}
             io:=ioresult;

             IF (io = 0) AND (pos('#',Ligne)=0)
             THEN writeln(ligne)
             ELSE NoCardinal:= False;
     END;
END;
(* - - - - - - - - - - FIN DE AffTexte - - - - - - - - - - - - - - - - - - *)

Procedure NextCardnl ( Var  io: integer;  VAR f : text ; Var ligne : StPhr);

{ Pre-cond : LIGNE la derniere ligne lue a partir de F.
  argt     : f,ligne
  reslt    : io, ligne
  Post-cond: IO possede le code retour des entrees/sorties pour F.
             Si io = 0 alors
                si not(eof(f)) ligne est premiere ligne suivante de F
                contenant un #
                SINON Ligne est la derniere de F
}

BEGIN
     WHILE (io=0) AND (Not(eof(f))) AND (Pos('#',ligne) = 0)  DO
     BEGIN
          {$i-}  readln(f,ligne); {$i+}
          io:=ioresult;
     END;
END;
(* - - - - - - - - - - -- Fin de NextCardnl - - - - - - - - - - - - - - - - *)

Procedure RecoitReponse (VAR   Answer : integer; VAR Cont : Boolean;
                               SetOfAnswer : SOA);

(* SetOfAnswer : ensemble de solutions possibles .
   Cont        : FALSE Si demande d'interruption .
   Answer      : Reponse De l'utilisateur si CONT = TRUE.

ARGT : SetOfAnswer
Reslt: cont, answer
Action : Lit la reponse de l'utilisateur, et la met dans Answer
*)

VAR    oui,reponse : SOA;
       Confirm     : Boolean;
```

```
      message      : string  ;
      confirmation : integer;


BEGIN
  confirm:= false;
  oui:= [ord('O'), ord('o')];
  reponse:= oui + [ord('N'), ord('n')];


 { Lire la reponse.}
  WHILE (cont) AND (not confirm) DO
  BEGIN
      message:='ESC : Interrompre';
      FOR i:= 1 TO 45 DO message:= Message + ' ';

     message:= message + 'Votre r{ponse : ';
     InVaffiche(1,25, message);
     lireReponse(SetOfAnswer, Answer);

     IF ( Answer = 27 )    { Demande d'interruption .}
     THEN cont:= false
     ELSE BEGIN
          Message:= 'ESC : Interrompre';
          For i:= 1 TO 18 DO message:= message+' ';
          message:= message + '^ est votre r{ponse: Confirmez (O)ui/ (N)on';
          message[pos('^',message)]:= chr(Answer);
          InvAffiche(1,25,message);
          Gotoxy(79,25);
          LireReponse(Reponse , confirmation);
          gotoxy(79,25);
          write(' ');

          IF (Confirmation = 27)
          THEN cont:= false
          ELSE confirm:= (confirmation IN oui);
      END;
  END;
END;
(* - - - - - - - - - - -  fin de RecoitReponse - - - - - - - - - - - - - - - *)



 { DEBUT DE QUESTION }

 BEGIN
      clrscr;

      totalSigne:= 0;
      totalquest:= 0;
      BonneReponse:= 0;
      totalseconde:= 0;
      cont:= false;

      assign(f,name);
      {$i-} reset(f); {$i+}
      io:= ioresult;
      more:= not(eof(f));

      IF (more) AND (io = 0)
      THEN BEGIN
           {$i-}  readln(f,ligne); {$i+}
           io:= ioresult;
           IF (io = 0)
           THEN BEGIN
```

```
          PasQuestion:= ( pos('#', ligne) = 0 );
          IF (PasQuestion)
     (    THEN BEGIN
               StClig:='B.E. : Pour Commencer @ lire';
               message:= 'B.E. : Pour Commencer @ lire';
               FOR i:= 1 TO 34 DO message:= message+' ';
               Message:= message + 'ESC : Interrompre';

               ecranbas(StClig, Message, 1, 25);

               Message:= 'B.E. : Poursuivre ';
               FOR i:= 1 TO 43 DO message:= message+' ';
               Message:= message + 'ESC : Interrompre';

               cont:= encore;
          END;
     END;
END;

WHILE (io = 0) AND (more) AND (PasQuestion) AND (cont) DO
BEGIN
     clrscr;
     NrLigne:= 0;
     WHILE (io = 0) AND (more) AND (PasQuestion)
                    AND (NrLigne < MaxNrLigne )  DO
     BEGIN
          NrLigne:= NrLigne+1;
          TotalSigne:= TotalSigne + CombienSigne(ligne);
          writeln(ligne);
          more:= not(eof(f));
          IF (more)
          THEN BEGIN
               {$i-} readln(f,ligne); {$i+}
               io:= ioresult;
               IF (io = 0)
               THEN BEGIN
                    PasQuestion:= ( pos('#', ligne) = 0);
               END;
          END;
     END;
     IF (io = 0) THEN
     BEGIN
          InvAffiche(1, 25, message);
          Date(jour1);
          temps(hor11);
          cont:= encore;
          Date(jour2);
          temps(hor12);
          Difftemps(jour1, jour2, hor11, hor12, result, seconde);
          TotalSeconde:= TotalSeconde + Seconde;
     END;
END;

{ S'occuper des questions.}

WHILE (not(eof(f)))  AND (io = 0) AND (Cont) DO
BEGIN
     TrouvQuest(io, f, correct, ligne, WaitLigne);
     IF (io = 0) AND (correct) THEN
     BEGIN
          TotalQuest := TotalQuest+1;
          TrouvSolution(ligne, solution, SetOfAnswer);
          ligne:= WaitLigne;
          AffTexte( f, io, ligne);
          IF (io = 0)
          THEN BEGIN
```

```
                    RecoitReponse(Answer, cont, SetOfAnswer);
                    IF (cont) THEN
                    BEGIN
                         IF (answer = solution)
                         THEN BonneReponse:= BonneReponse + 1;
                         NextCardnl(io, f, ligne);
                    END;
               END;
          END;
     END;
     Close(f);
END;

(* - - - - - - - - - - - -  fin de Question - - - - - - - - - - - - - - - - *)


Procedure Forme (Depart : PtR3St;
                 var duree : integer;
                 VAR result : TI;
                 VAR NoBreak : Boolean
                  );


{PRE-Cond : DEPART pointe sur une chaine .
ARGT      : Depart
Reslt     : Duree,Result, Nobreak
Post-Cond:
          NoBreak = True, s'il n'y a pas eut de demande d'interruption;
                  = False sinon.
           RESULT(jour mois annees) ,DUREE (en secondes) sont  les temps
           de lecture des elements de la chaine.


}
CONST   leg = 2 ; { indice de ligne de la legende.}
       prem = 4; { indice de la premiere ligne .}

VAR                                mesg, mesclig : st80   ;
                                       nombre, i : integer;
                                       eol, cont : boolean;
              jour1, jour2, hor11, hor12, tab3 : TI      ;
                                         legende : stPhr  ;
                                             R3: R3st     ;

BEGIN

     Clrscr;

     InitList3(depart,eol);
     cont:= not(eol);

     IF (cont) THEN
     BEGIN
          legende:= ' B.E. : Declencher
  ESC : Interrompre ';
          invaffiche(1, leg, legende);
          nobreak:= encore;
          cont:= nobreak;
          legende:= ' B.E. : Poursuivre
  ESC : Interrompre ';
          invaffiche(1, leg, legende);
     END;
     { Prendre l'heure de depart.}
     date(jour1);
     temps(hor11);
```

```
    WHILE CONT DO
    BEGIN
         nextlect3(depart,eol,R3);
         cont:= not(eol);

         { Afficher le mot a reperer.}

         gotoxy(1, prem);
         writeln('-) ',R3.ORG);
         NoBreak:= Encore;
         cont:= cont AND NoBreak;
         gotoxy(1, prem);
         clreol;

    { Afficher les lignes.}
         IF (NoBreak) THEN
         BEGIN
              FOR i := 1 TO 3 DO
              BEGIN
                   gotoxy(1,Prem+i);
                   write(R3.lgn[i]);
              END;
              noBreak:= Encore;
              cont:= CONT AND NoBreak;

              { EFFACER LES LIGNES.}

              FOR i := 1 TO 3 DO
              BEGIN
                   gotoxy(1,Prem+i);
                   clreol;
              END;
          END;
    END;

    { Eteindre le chrono.}
    temps(hor12);
    date(jour2);

    { Calculer le temps. }
    difftemps(jour1, jour2, hor11, hor12, result,duree);


END;
(* - - - - - - - - - - - - fin de FORME - - - - - - - - - - - - - - - - - - *)

Procedure Regression(Depart : PtStPh;
                     var duree,regress,qte : integer;
                     VAR result : TI;
                     VAR NoBreak : Boolean
                     );

{ ARGT : Depart
  Reslt: Duree, regress, qte, result, nobreak

  Si depart pointe sur une chaine contenant des phrases d'exercices pour la
  regression, Result (En heure , min, sec ) , DUREE (en secondes)  designe le
  temps necessaire pour lire QTE Phrases avec REGRESS regressions.
  Si NOBREAK = False, le lecteur avait demande une interruption les autres
  resultats sont indefinis.
}


CONST     Last = 24;  { Indice De La Derniere Ligne.}
          prem = 3 ;  { Indice de la 1ere ligne .}
```

```
VAR          Legende,Phrase : StPhr;
                 nombre, i  : integer;
                 eol, cont  : boolean;
jour1, jour2, horl1, horl2  : TI      ;


BEGIN

    Clrscr;
    duree:= 0;
    qte:=0;
    nobreak:= true;
    regress:= 0;                                                    E

    Legende := ' B.E. : Declencher
C : Interrompre';

    INVaffiche(1,1,Legende);

    nobreak:= encore;

    IF (NoBreak) THEN
    BEGIN
        Legende := ' B.E. : Poursuivre                    (ü : regression
   ESC : Interrompre';
        INVaffiche(1,1,Legende);

       initlistP(depart,eol);
       cont:= not(eol) AND NoBreak;

       { Prendre l'heure de depart.}
       date(jour1);
       temps(horl1);

       WHILE CONT DO
       BEGIN
           i:= prem;
           WHILE(cont) AND (i <= LAST) DO
           BEGIN
                nextlectP(depart,eol,phrase);
                gotoxy(1,i);
                clreol;
                writeln(phrase) ;
                qte:= qte+1;
                i:= i+2;

           END;

           { Effacer Lignes restantes.}

           WHILE(cont) AND (i <= LAST) DO
           BEGIN
                 gotoxy(1,i);
                 clrEol;
                 i:=i+2
           END;

           encoreEnter(NoBreak,Nombre);
           Regress:=Regress+nombre;
           cont:= not(eol) AND NOBREAK;

       END;

       { Eteindre le chrono.}
       temps(horl2);
```

```
      { Calculer le temps. }
      difftemps(jour1, jour2, hor11, hor12, result,duree);

   END;
END;
(* - - - - - - - - - - - fin de Regression - - - - - - - - - - - - - - - - - - *)


Procedure NomFich(Depart : PtStMt;    VAR StCour: StMot; VAR ESC : boolean);

{ PRE-CONDITIONS : Depart pointe sur une caine non vide de strings mot.
  ARGTS          : Depart;
  RESLT          : STCOUR, ESC
  POST-CONDITION : S'il y a eut interruption ( ESC = TRUE),
                   alors StCour est Vide
                   SINON STCOUR contient le mot choisi par l'utilisateur.
  ACTION         : Nomfich affiche a l'ecran les choix possibles.
}

VAR   val, qte,
      colcour, indcour, lgncour   : integer;
                         ligne    : st80;
                           tab    : TabMot;
                cont, suiv, prec  : boolean;
                        courant   : PtStMt;


Procedure TrtESC;

BEGIN           { Interruption.     }
     cont:= false;
     ESC := False;
     StCour:= '';
END;
(* - - - - - - - - - - - - - - - fin de TrtEsc  - - - - - - - - - - - - - - - *)

Procedure trtHaut;
BEGIN           (* Haut.                *)
     centrage(lgncour, colcour, StCour);
     IF (IndCour > 1)
     THEN  BEGIN
           indcour:= indcour-1;
           IF (lgnCour = 6)
           THEN BEGIN
                lgnCour:= 20;
                colcour:= 21;
           END
           ELSE  lgnCour:= lgnCour - 2;
      END
      ELSE BEGIN
           IF (prec)
           THEN BEGIN
                prvPt(courant);
                FillTab( courant, tab, qte, suiv, prec);
                AffTab(Tab);
                indcour:= qte;
                lgnCour:= 20;
                ColCour:= 58;
           END;
      END;
      StCour:= Tab[IndCour]
END;
```

```
Procedure TrtPGUP;
BEGIN          { PG UP.              }
     centrage(lgncour, colcour, StCour);
     IF (prec)
     THEN BEGIN
          prvPt(courant);
          FillTab( courant, tab, qte, suiv, prec);
          AffTab(Tab);
          indcour:= qte;
          lgnCour:= 20;
          ColCour:= 58;
     END;
     StCour:= Tab[Indcour];
 END;
(* - - - - - - - - - - - - - fin de TrtPgUp -- - - - - - - - - - - - - *)

Procedure TrtGauche;
BEGIN          { <-                  }
     centrage(lgncour, colcour, StCour);
     IF (indcour > 8)
     THEN BEGIN
          indcour:= indcour-8;
          ColCour:= 21;
     END;
     StCour:= Tab[indcour];
 END;
(* - - - - - - - - -  fin de TrtGauche - - - - - - - - - - - - - - - - - - *)

Procedure trtDroite;
BEGIN    {  ->     }
     centrage(lgncour, colcour,StCour);
     IF (indCour + 8 <= Qte)
     THEN BEGIN
          indcour:= indcour+8;
          colcour:= 58
     END;
     StCour:= Tab[indCour];
 END;

(* - - - - - - - - - - - - - - Fin de TrtDroite - - - - - - - - - - - - - - *)
Procedure trtBas;
BEGIN          { Bas.                }
     centrage(lgncour, colcour, StCour);
     IF (IndCour < Qte)
     THEN  BEGIN
           indcour:= indcour+1;
           IF (lgnCour = 20)
           THEN BEGIN
                lgnCour:= 6;
                colcour:= 58;
           END
           ELSE  lgnCour:= lgnCour + 2;
     END
     ELSE BEGIN
          IF (Suiv)
          THEN BEGIN
               NxtPt(courant);
               FillTab( courant, tab, qte, suiv, prec);
               AffTab(Tab);
               indcour:= 1;
               lgnCour:= 6;
               ColCour:= 21;
          END;
     END;
```

```
     StCour:= Tab[Indcour]
END;
(* - - - - - - - - - - - - -  fin de TrtBas - - - - - - - - - - - - - - - - *)

Procedure trtPgOn;
BEGIN       { PG ON.        }
     centrage(lgncour, colcour, StCour);
     IF (suiv)
     THEN BEGIN
          NxtPt(courant);
          FillTab( courant, tab, qte, suiv, prec);
          AffTab(Tab);
          Indcour:= 1;
          lgnCour:= 6;
          ColCour:= 21;
      END;
      StCour:= Tab[Indcour];
 END;
(* - - - - - - - - -  fin de  TrtPgOn - - - - - - - - - - - - - - - - - - *)

{Debut De  NomFich. }

BEGIN
  courant:= depart;
  filltab(courant, tab, qte, suiv, prec);

  IF (QTE <= 0 )
  THEN stcour:=''
  ELSE BEGIN

      Clrscr;
      cadre(18,62,1,3);
      gotoxy(20,2);
      write('CHOISISSEZ LE TEXTE QUI VOUS INTERESSE !');

      traceligne(40, 5, 18, '£', true);
      cadre(2, 78, 4, 22);

      ligne:= '  d : Droite   £  h : Haut  £  PG UP : Ecran Pr{c{dent  £  ESC : I
 nterrompre ';
      ligne[pos('d', ligne)]:= chr(26);
      ligne[pos('h', ligne)]:= chr(24);
      invaffiche(2, 24, ligne);

      ligne:= '  g : Gauche   £  b : Bas   £  PG ON : Ecran Suivant    £  {ü  : {
 hoisir     ';
      ligne[pos('g', ligne)]:= chr(27);
      ligne[pos('b', ligne)]:= chr(25);
      invaffiche(2, 25, ligne);

      colcour:= 21;
      lgncour:= 6;
      IndCour:= 1 ;
      StCour := tab[indCour];
      cont:= true;
      ESC:= False;

      AffTab(tab);

      WHILE (cont) DO
      BEGIN
           inversv;
           centrage(lgncour, colcour, Tab[Indcour]);
           normalv;
           coin;
```

```
        effacecurseur;
        normalv;
        accept(val);
        CASE val OF
             13 :  cont:= false;     { String est choisi.}
             27 :  TrtESC;           { Interruption.      }
             99 :  TrtHaut;          {  Haut.             }
            100 :  TrtPGUP;          { PG UP.             }
            102 :  TrtGauche;        { (-                 }
            104 :  TrtDroite;        {  -)                }
            107 :  trtBas;           { Bas.               }
            108 :  TrtPgOn;          { PG ON.             }
        END;
    END;
  END;
END;
(* - - - - - - - - - - - - - - fin de NomFich - - - - - - - - - - - - - - - *)


Procedure ShowY(VAR NoBreak: boolean; VAR result, result2 : TI;
                VAR duree, duree2, NbrCourt, Nbrlong : integer;
                VAR f: frec);
{ PRE-COND   : F contient les lignes d'un texte en Y.
  ARGT       : F
  RESULTAT   : Nobreak, result, result2, duree, duree2, NBRCOURT, NBRLONG.
  POST-COND  : SI NOBREAK = False le lecteur a interrompu
               SINON RESULT du type Heure, min, sec et DUREE(en secondes)
                     est le temps necessaire a lire les NBCOURTS premieres
                     phrases de F
                     RESULT2 du type Heure, min, sec et DUREE2(en secondes)
                     est le temps necessaire a lire les NBRLONG phrases de F.
                     F contient NBRCOURT + NBRLONG Phrases.
}

CONST
     PremLgn = 4;
     lnLegende = 2;
        colMil = 40;
VAR
   nbreLu,i,j,ln,lc, codec, nbrelement : integer;
   legende : StPhr;
   cont    : boolean;
         r : rec;
   jour1,hor11 , jour2,hor12: ti;
BEGIN
     clrscr;
     NbrElement:=0;
     NbreLu:= 0;

     LEGENDE:=' B.E.: POURSUIVRE';                                          ';
     Legende := Legende + '
     legende := legende + 'ESC : INTERROMPRE ';
     RESET(f);
     cont:= not(eof(f));

     {Afficher l'ecran}
     IF cont THEN
     BEGIN
          read(f,r);
          VAL(R.elm, nbrCourt, i);
          InvAffiche(1,LnLegende,legende);
          noBreak:= encore;
          cont:= not(eof(f)) and nobreak;
          IF (cont) THEN Read(f,r);
          traceligne(1,premlgn,MaxCol,'¥',false);
          i:= PremLgn + 1;
```

```
        j:=1;
        ln:=0;

        {Declencher le Chrono.}
        date(jour1);    Temps(hor11);

        {Afficher a l'ecran}
        WHILE (cont) DO
        BEGIN
             gotoxy(1,j);
             delLine; delLine;

             {Afficher tant qu'on a pas lu tout le fichier}
             WHILE (cont)  AND (i <= MaxLgn) DO
             BEGIN



                  ln:=ln+1;
                  Gotoxy(1,i);
                  IF (LN <= NbLn ) THEN
                  BEGIN
                       centrage(i, ColMil, r.elm);
                       lc:= ColMil- (length(R.elm) DIV 2)
                  END
                  ELSE BEGIN
                       gotoxy(lc,i);
                       write(r.elm);
                  END;
                  IF(Ln = (2*NbLn)) THEN ln:=0;

                  cont:= not(eof(f));
                  IF (cont) THEN read(f,r);
                  i:=i+2;
                  NbrElement:= nbrElement+1;
             END;
             i:=i-2;
             noBreak:=encore;
             nbreLu:=Nbrelu+1;
             BonTpsPfc(jour1, Hor11, result,duree,codec, nbrcourt, NbreLu);
             cont:= cont AND noBreak AND (codec = 0);
        END;

        {Faire monter les lignes deja affichees}
        WHILE (Nbrelu < NbrElement) AND noBreak AND (Codec = 0) DO
        BEGIN
             GOTOXY(1,j);
             delLine; DelLine;
             NoBreak:= encore;
             nbreLu:= NbreLu+1;
        END;
        date(Jour2);
        Temps(Hor12);
        DiffTemps(jour1, jour2, Hor11,Hor12, result2, duree2);
        nbrLong:= NbrElement - NbrCourt;
    END;
END;

{--------------------    fin de ShowY - - - - - - - - - - - - - - - - -----}
```