MASTER EN SCIENCES INFORMATIQUES

Pour un projet d'enseignement assisté par ordinateur sur l'accord du verbe

Gouthière, David

Award date:
1984

Awarding institution:
Universite de Namur

POUR UN PROJET D'ENSEIGNEMENT ASSISTE PAR ORDINATEUR SUR L'ACCORD DU VERBE.

PRESENTE PAR DAVID GOUTHIERE

PROMOTEUR:MONSIEUR CLAUDE CHERTON

FACULTES UNIVERSITAIRES N.D. DE LA PAIX NAMUR

INSTITUT D'INFORMATIQUE
rue Grandgagnage, 21, b - 5000 Namur (Belgique)

POUR UN PROJET D'ENSEIGNEMENT ASSISTE PAR ORDINATEUR SUR L'ACCORD DU VERBE.

Ce travail comprend

. L'analyse d'une démarche pédagogique

. Les bases de réalisation du projet

- Correction automatique de fautes de frappe et de fautes d'orthographe d'usage
- Base de données et génération de phrases.

présenté par David Gouthière

promoteur: Monsieur Claude Cherton

Merci à Monsieur Joseph Tonneau et à Monsieur René Wéry, votre expérience est précieuse pour ce qui touche à la pédagogie dans mon mémoire. Merci pour votre dévouement.

Merci à Monsieur Claude Cherton pour vos bons conseils et surtout pour la grande liberté que vous m'avez laissée tout au long de l'année.

TABLE DES MATIERES

Le présent travail se compose de quatre chapitres.

Le choix du sujet
L'analyse
L'implémentation
Les conclusions

I. LE CHOIX DU SUJET

Je vous présenterai les usages que l'enseignement en tant que tel peut faire de l'ordinateur. Je donnerai quelques objectifs de l'EAO, je formulerai des critiques. Enfin je vous dirai comment et pourquoi j'ai choisi ce sujet.

II. L'ANALYSE

Ce chapitre vous présentera successivement la définition d'une démarche pédagogique et l'élaboration de sa réalisation. Vous verrez naître les nécessités de créer automatiquement des phrases et de pouvoir décoder les phrases d'un enfant. Vous découvrirez les difficultés que cela provoque ainsi que les solutions informatiques imaginées pour les résoudre.

III. L'IMPLEMENTATION

Ce chapitre présentera uniquement les techniques employées pour implémenter la base de données,en mémoire centrale et sur disquette. Le reste me semble moins intéressant et figurera plutôt en annexe.

Je vous récapitulerai ce que j'ai fait et pas fait, je tirerai des leçons et je vous dirai que j'aimerais poursuivre ce projet.

***

PREMIERE PARTIE : LE CHOIX DU SUJET.

A. INTRODUCTION A L'EAO

B. OBJECTIFS INTERESSANTS EN EAO

C. POURQUOI AI-JE CHOISI UN MEMOIRE EN EAO ?

D. COMMENT S'EST FAIT LE CHOIX DU SUJET ?

I. LE CHOIX DU SUJET.

A. Introduction à l'EAO, Enseignement Assisté par Ordinateur.

1. Présentation.

Avant tout, et pour éviter toute ambiguité, je dirai que je désigne par EAO tous les usages que l'enseignement en temps que tel peut faire de l'ordinateur.

On distingue aujourd'hui trois types d'usages.

a. l'apprentissage d'une nouvelle matière.

L'élève est confronté de façon active, il y a dialogue avec la machine, ou passive, il n'ya pas dialogue, à une matière qu'il n'a pas vue au cours.

b. perfectionnement dans une matière vue au cours.

L'enfant a appris quelque chose au cours.
Un programme informatique lui illustre alors la matière sous un angle nouveau ou lui propose des exercices.
Le but est de faire acquérir à l'enfant ce qui lui aurait échappé ou/ et de lui faire atteindre un niveau de détail plus profond.

Ce sont les programmes de simulation, les exercices de rattrapage.

c. formation de l'esprit.

Le but est de faire acquérir à l'enfant certains modes de réflexion indépendamment du programme des cours.
Par exemple la notion de récurssivité (je pense au système Logo) ou la démarche scientifique.

2. Remarque.

Cette classification très rigide reflète bien la jeunesse de l'EAO. Les horizons sont multiples. Pour clarifier, la tendance est alors de diviser le morceau.

C'est en gros la classification ci-après.

Heureusement, ceci n'est pas une fin en soi !

Arrêtez votre lecture et imaginez quelques instants un maître donnant la leçon, un problème de robinet qui fuit, par exemple ...

" le maître lit l'énoncé, le fait écrire, le décortique, le
" précise.
" Il interroge, s'assure que chacun l'a lu.
" Il revient à une leçon précédente, rappelle un point oublié,
" fait le lien. Un de ses petits sujets le questionne ? Le
" maître lui répond et, au besoin ouvre une parenthèse. Il
" demande aux enfants leurs idées, les écrit au tableau, les
" discute - qui n'est pas d'accord ?, qui est d'accord ?
" Il retient les bonnes idées, éloigne les mauvaises, parfois,
" il en garde une mauvaise pour épier les réactions etc ...".

Dans cette leçon, le maître inculque d'abord une méthode de résolution, une démarche scientifique (énoncé, idées de solution, sélection mise en oeuvre, constatations etc ...)

Plus explicitement, il fait de l'apprentissage et du rattrapage (il rappelle une formule oubliée).

Nous retrouvons donc les 3 points de la classification complètement intégrés dans une méthode pédagogique.

Je ne discuterai pas de la méthode assez directive employée. Je veux plutôt souligner la richesse du dialogue et admirer la diversité des comportements que prend le maître pour répondre aux différentes difficultés de ses élèves.

Il me semble intéressant qu'un didacticiel dialogue lui aussi avec l'enfant, mais il n'est pas nécessaire quece dialogue soit du même type que celui entre le maître et son élève.

En effet, pourquoi s'obstiner à imiter en moins bien ce qui existe déjà. Notre but est d'aider le professeur avec un outil nouveau, pas de le remplacer.

Il faudrait donc imaginer, selon l'objectif du didacticiel un système de dialogue propre à l'ordinateur et qui utilise ses avantages.
Ce qu'on appelle dialogue n'est pas nécessairement un échange de paroles ou de textes en français, mais correspond plus globalement à échange d'information.
Le fait de montrer un dessin, de réaliser un exercice est aussi un transfert d'information qui fera réagir l'ordinateur (Comment ?)

Ce système serait d'un niveau suffisamment élevé pour débusquer les difficultés des enfants.

Evidemment, la qualité du dialogue serait fonction de ce que le pédagogue/informaticien aurait mis dans le programme.
Elle se marque par la pertinence des exercices offerts, par les possibilités de réponses laissées à l'enfant, par l'interprétation qu'en fait le programmeur, via son programme, par la qualité et la diversité des explications qu'il donne, par la qualité de la démarche pédagogique.

(Une anecdote : mon instituteur-conseil me suggère de ne pas dire "le sujet est derrière le verbe (ou devant) mais dire "le sujet est après le verbe (ou avant) parce qu'il a remarqué que c'était plus "parlant").

3. Quelles sont les critiques que l'on peut formuler à l'égard de l'EAO ?

a. le manque d'organisation.

Parmi les applications peu nombreuses existant aujourd'hui en Belgique, je distingue :

- **des programmes commerciaux.**

  Ils sont généralement produits par des firmes (quand elles s'intéressent à l'EAO).
  Pour séduire le client, elles vantent leur technologie chatoyante (écrans couleurs à haute définition, très jolis dessins) et une ergonomie très poussée. Ces programmes sont très faciles à utiliser et extrèmement gais.

  Ces firmes se taisent généralement sur la valeur pédagogique très discutable de leurs produits.
  J'ai vu à Bruxelles dans un salon consacré à l'EAO un constructeur japonais qui proposait notamment un jeu de pendu pour apprendre l'orthographe !

  Un micro ordinateur me semble un peu "chérot" pour ce genre de pédagogie.

- **des programmes à vocation réellement pédagogique.**

  Ils sont conçus par des chercheurs qui aiment à la fois l'informatique et la pédagogie. Ils voudraient concilier les deux. Ces chercheurs sont généralement des enseignants et des informaticiens (ou les deux). Ils sont plus ou moins spécialisés en informatique et en pédagogie selon leur formation initiale.

  Il y a malheureusement un manque de communication entre les gens et les firmes qui réalisent tous ces didacticiels.
  J'estime que les idées, qu'elles soient techniques ou pédagogiques, devraient circuler beaucoup plus (cours d'informatique aux enseignants désireux, publication de propositions de didacticiels et des résultats des expériences déjà menées etc...)

Il est extrèmement navrant de voir que des gens qui dominent bien la programmation écrivent des programmes bons à jeter parce qu'il leur manque les conseils d'un enseignant averti.

Il est tout aussi désolant de penser que des idées géniales ne se concrétisent pas parce que personne ne sait qu'elles existent et que les novateurs ne disposent pas des moyens financiers ou des connaissances techniques suffisantes.

Attention, je ne voudrais pas que l'on croie qu'il n'existe rien et que partant chacun travaille dans son petit coin. Il existe d'excellents didacticiels qui ont fait l'objet d'expériences largement publiées. Logo est là pour m'appuyer.

b. L'EAO a l'avantage d'exister.

N'oublions pas les avantages inhérents à l'utilisation de l'ordinateur.

Il peut être utilisé individuellement; il peut être programmé très patient pour encourager l'enfant qui aurait peur du maître.

Mais plus fondamentalement, je pense que confronter très tôt l'enfant à l'informatique luipermettra d'accepter le mode de vie qu'il aura dix ans plus tard.

J'estime essentiel, sur le plan social, que la future génération n'ait plus les craintes des gens d'aujourd'hui vis à vis d'un outil qu'ils ne comprennent pas ou peu.

B. Objectifs intéressants en E A O.

1. créer une communication informaticien-enseignant.

Je pense que pour tirer le maximum d'un outil couteux, il serait bon de faire coopérer les informaticiens et les enseignants.

Ceci permettrait de concevoir des didacticiels de haut niveau pédagogique soutenus par un bon système informatique souple, agréable et fiable.

C'est notamment ce qui se fait à Namur dans le cadre du CEFIS (centre pour la formation à l'informatique dans le secondaire) et de plusieurs mémoires de fin d'études chez Monsieur Cherton.

Tout comme il existe des initiations des enseignants à l'informatique, il serait également intéressant d'initier des informaticiens à la pédagogie.

2. développer l'EAO dans des branches moins traditionnelles.

On comprend que les mathématiciens, naturellement attirés par un produit de la recherche scientifique, aient été les premiers utilisateurs dans l'enseignement.

Mais il faut comprendre aussi que des domaines moins traditionnels comme le français, les langues étrangères, l'économie, l'histoire et autres, peuvent donner lieu à des modèles tout aussi formalisables qu'un modèle mathématique.

Voici par exemple les problèmes grammaticaux, l'évolution des prix sur le marché ...

Ces domaines peuvent donc donner lieu à des programmes d'ordinateurs qu'il serait intéressant d'expérimenter en EAO.

C. Pourquoi ai-je choisi un mémoire en E A O ?

Je cherchais surtout un sujet qui m'amuse et que je puisse mener seul.
L'idée de travailler pour d'autres utilisateurs (enfants, enseignants) ayant d'autres besoins (outil pédagogique), m'a séduit.

L'EAO me permettait également de me "frotter" aux problèmes de langue naturelle si je choisissais un domaine littéraire.

Enfin, je dois dire que c'est très motivant de réaliser quelque chose d'utile qui servira.

D. Comment s'est fait le choix du sujet ?

Lors d'une réunion avec plusieurs membres du corps enseignant, j'ai pu formuler quelques idées et mon désir de réaliser un programme d'EAO dans la discipline du français et plus particulièrement l'orthographe pour des exercices de rattrapage et du niveau primaire.

Deux professeurs intéressés ont proposé de me conseiller.
Il s'agit de Monsieur WERY, instituteur à Gembloux et Monsieur TONNEAU, Inspecteur de l'enseignement.

Nous nous sommes entendus sur le problème de l'accord du verbe avec son sujet, gros point noir chez leurs élèves.

SECONDE PARTIE : L'ANALYSE

A. DEFINIR LE PROJET

1. Définir une démarche pédagogique
2. Spécifications du problème

B. ELABORER UNE SOLUTION INFORMATIQUE

1. Introduction
2. Définition des données
3. Définition des grands traitements
4. Structure de la base de données
5. Les deux applications existantes
   - La correction automatique des fautes d'orthographe d'usage et des fautes de frappe
   - La génération de phrases

II. ANALYSE.

A. Définir le projet.
B. Elaborer une solution informatique.

A. Définir le projet.

1. Définir une démarche pédagogique.

Pour avoir un bon résultat, chaque étudiant de Monsieur Cherton travaillait avec un ou plusieurs enseignants.

Monsieur René WERY, instituteur, et Monsieur Joseph TONNEAU, inspecteur, m'ont guidé et m'ont donné les bases pédagogiques nécessaires.

Après quelques heures de discussion, nous nous sommes arrêtés sur les trois points suivants.
La notion de référentiel; demander à l'enfant le maximum d'initiative, adapter les exercices aux faiblesses propres de l'enfant et suivre son évolution.

a. notion de référentiel.

1) définition .

Un référentiel est un ensemble de références composé de règles et d'exemples types.
Chaque exemple est l'illustration d'une règle connue du possesseur du référentiel.

La règle peut être exacte : son application donne toujours un résultat correct.
bonne : son application donne un résultat correct dans les cas courants.
(NB) fausse : son application donne des résultats abérrants.

Ainsi, quand l'enfant doit orthographier un mot, régler un accord, il se réfère à un cas qu'il estime semblable et qu'il connaît. Il lui reste à appliquer à son mot ou à son accord la règle qui est appliquée à son exemple type.

(NB) je parlerai plus généralement de "référentiel exact" "bon référentiel" "réfenrentiel faux"

Ex. l'enfant écrit :

je les donnes

il choisit donc :

il devrait choisir :

dans son référentiel il a en outre :

règle		ex. type
les .. "s"		les pommes
ind.présent ER		
je	e	je frappe
tu	es	tu frappes
il	e	il frappe
nous	ons	nous frappons
vous	ez	vous frappez
ils	ent	ils frappent

2) analyse du problème.

Le problème se pose à 2 niveaux.

soit il n'a pas de référence pour le cas qui se pose, parce qu'il n'a pas vu la matière, ne l'a pas comprise ou l'a oubliée.

Je pense qu'il s'en sortira peut-être en demandant au professeur et plus probablement en appliquant une règle non adaptée, comme la très naturelle règle phonétique.

Ceci est très dangereux, car il s'ancre lui-même des erreurs dans la tête.

soit, il possède des références, mais dont les règles sont fausses. Comme souvent les règles fausses sont plus simples à appliquer que les bonnes et les exactes, il les choisit.

exemples de règles fausses.

. règle phonétique.
je prononce pareil, j'écris pareil ou j'écris comme je l'entends.
Cela donne par exemple :
"de l'ont bras" pour de "longs bras"

. règle trop large.
" après "les", je mets "s".

cela donne par exemple : "je les manges" pour "je les mange".

exemple de bonnes règles.

cf. Pratique de la langue De Meur-Muset - 6è année - éd. De Boeck
P 128 - tableau de Massarenti.
découpe des verbes connus en verbes en E-IS-S.

(photocopie en annexe)

il s'agira donc de créer chez l'enfant un bon référentiel, ceci en supprimant les règles fausses qui sont ancrées dans son esprit et en y faisant entrer de "bonnes règles et des règles exactes".

b. permettre à l'enfant le maximum de créativité.

Dire dans un article sur la pédagogie "je veux permettre à l'enfant le maximum de créativité", c'est à coup sûr m'attirer la sympathie de pas mal d'enseignants et psycho-pédagogues.

Attention messieurs, il y a un piège !

Si vous le voulez bien, penchons-nous ensembles quelques secondes sur le mot créativité.
C'est un mot très flatteur : il suggère l'invention, l'originalité l'imagination, l'épanouissement, la libre expression ...
Méfions-nous, dans la réalité, ce n'est pas toujours si rose.

Il y a toutes les contraintes extérieures comme le contexte culturel, politique, économique, social qui font qu'une personne qui vit en Irlande ne créerait pas la même chose si elle vivait en Sicile. Il y a des contraintes que l'on ressent plus directement, comme par exemple celle de son chef de service qui a des idées que l'on juge étroites, ou un manque de matériel ou encore des règles très strictes comme celles des rimes, des nombres de syllabes pour un poête.

Dans notre didacticiel, l'enfant devra obligatoirement répondre aux questions qu'on lui pose. Il ne pourra pas intervenir sans qu'on l'y invite.

Le seul choix qubn lui laisse est le contenu des phrases qu'il peut inventer, encore ne peut-il utiliser que le vocabulaire et les structures de phrases permises.

J'espère malgré tout que les didacticiels seront prévus pour permettre à l'enfant d'être curieux, pour l'inciter à poser lui-même des questions. Certains types d'applications, comme les simulations s'y prêtent très bien.
L'enfant peut se demander "que se passe-t-il si ... ?" et il peut jouer avec les paramètres.

Dans des exercices de rattrapage sur l'accord du verbe, cela me paraît plus difficile. J'attends vos idées.

c. suivi de l'enfant pour lui proposer des matières adaptées à son niveau.

L'image de l'école du début du siècle est celle du bon élève qui récite sa leçon tandis que le "bonnet d'âne" va au coin sous la risée des autres.

Ce tableau reflète le régime politique de l'époque.
Le principe de sélection favorisait les mieux nantis (conditions familiales aisées et propices aux études).
Il oubliait les autres et engendrait une élite dominante.
C'était une reproduction automatique du régime en vigueur.

Je désapprouve beaucoup cette sélection élitiste. D'abord elle est injuste. Elle oublie que les mals nantis ont très peu d'occasions de s'exprimer. Ensuite, elle retient l'écolier sur des critères définis par une minorité (savoir le latin, les math...) Elle ne lui permet pas de montrer sa capacité réelle dans ses domaines privilégiés (imagination, habileté manuelle, mémoire, intelligence ...)

Aujourd'hui dans un système plus socialiste, cette tendance est inversée.

L'idée est de donner à chacun, même au plus défavorisé, une réelle chance de promotion sociale par l'instruction.
J'aime beaucoup ceci.

Un premier pas est donc effectué. La sélection "mal nantis/bien nantis" s'oriente maintenant selon un axe "peu doués/fort doués"

Ce que je crains c'est que pour aider les plus faibles à réduire l'écart, on ne freine les plus forts.

Faisons un voyage en train. Un train tiré par une locomotive noire, une belle locomotive avec des gros pistons, une cheminée et beaucoup de fumée et des wagons. Appelons-le "le train des écoliers". Il avance à une allure modérée. Malheureusement, certains voyageurs le ratent. Il passe trop vite.
D'autres y sont installés depuis longtemps. Ils ont déjà vu le paysage et se sont endormis. Les derniers font leur plus beau voyage, ils découvrent des régions avec des tunnels, des montagnes et des plaines.

Dans mon tableau, vous avez compris que les voyageurs qui ratent le train sont les écoliers les moins doués, les dormeurs sont les trop doués.

Je n'aimerais pas du tout que l'on aide les retardataires en réduisant l'allure de ma belle loco.
Mais je refuse aussi que l'on gâche le beau voyage des uns pour réveiller les quelques uns qui s'ennuient.

Je voudrais tout simplement qu'il y ait trois locos, des belles, des avec de la fumée comme dans Tintin, de vitesses différentes.

Certains me diront alors : "avec tes trois locos, il y aura des premiers et des derniers. Cela va créer des inégalités, ce n'est pas juste".

Je leur répondrai : "oui, il y aura des différences, parfoistrès grandes à l'arrivée, pas des inégalités et elles seront basées sur des critères peut être plus moraux que ceux de 1920".

Ce qui me semblerait injuste, vraiment injuste, ce serait que les plus doués ne prennent pas leurs responsabilités de plus doués et s'ennorgueillissent.
D'un point de vue technique, cela signifie que je devrai essayer de moduler la vitesse de rattrapage/apprentissage selon la force de l'élève. C'est très dur. Il faudra adapter les types d'exercices que je propose à l'élève, selon le type de fautes qu'il fait; ceci étant particulier à chaque élève.

2. Spécifications du problème.

A. hypothèse de départ.
b. délimiter la matière
c. appliquer la notion de référentiel.
d. ex. de scénario
e. contraintes semantiques.

-

a. hypothèse de départ (niveau requis, ambition de notre programme).

Ce didacticiel sera prévu pour servir à des enfants de dix à douze ans du niveau de la 5è ou 6è année primaire.

La matière envisagée sera l'accord du verbe avec le sujet. Je dis bien l'accord du verbe et pas la conjugaison.

Pour atteindre l'objectif "accord duverbe", certaines connaissances de base sont souhaitables, d'autres sont nécessaires.
D'une part, il est souhaitable que l'enfant ait vu aucours les règles d'accord du verbe. S'il ne les a pas vues, le programme doit être capable de les lui faire découvrir.

D'autre part, il serait très utile que l'enfant connaisse la conjugaison des verbes courants à l'indicatif. Je suppose d'ailleurs que l'enfant dispose devant lui d'un grand tableau rappelant les règles de conjugaison de ces verbes.

Notre ambition n'est pas d'obtenir de l'enfant qu'il fasse "zéro faute". Plus sagement, nous désirons simplement qu'il ait une bonne orthographe adaptée aux besoins réels de la vie de chaque jour.

b. délimiter la matière du problème envisagé.

Après douze ans, j'ai bien entendu oublié la classification des modes d'accord. J'ai donc retrouvé ma grammaire et un précis d'orthographe pour faire un recensement.

Monsieur Tonneau, Monsieur Wery et moi avons sélectionné les cas à envisager.

Selon nos hypothèses, nous n'avons choisi que les cas usuels.

Nous avons ainsi rejeté :

. sujet collectif (ex. une armée de servantes)
. deux sujets singuliers unis par ni ou par ou.
. plusieurs sujets résumés dans un seul mot (tout, rien ...)
. sujets en gradation (un seul mot, un soupir, un coup d'oeil nous trahit)
. plusieurs sujets représentant un seul être ou un seul objet. (ex. comme chaque matin, une mince colonne lilas, une tige de lumière, debout, divise l'obscurité de la chambre).
. le sujet est un adverbe de quantité.
. le sujet contient la locution " le peu de ".

pour conserver :

. le cas général : sujet suivi du verbe.
. l'inversion du sujet
. le sujet tu
. le sujet ON
. le sujet QUI.
. présence d'un écran entre le sujet et le verbe.

Avant de vous exposer comment nous appliquons la notion de référentiel, j'aimerais que vous cessiez un instsnt la lecture pour consulter ces quelques pages issues de dictées de petits enfants.

Cela vous aidera sûrement à comprendre le problème qui nous préoccupe.

HERPELINCK (Luc) i 18/8/185

S.L.

Conception d'un système d'aide à l'élaboration de devis / Luc Herpelinck. -

[S.l. : chez l'auteur] , 1984. -
1 vol. pag. mult. : ill. ; 30 cm. -
(Mémoire / Facultés univ. N.D. de la Paix, Namur. Informatique ; 1984)

90290

c. appliquer la notion de référentiel.

1. rappel.
2. créer un scénario
3. les variations.

1) rappel.

Je vous ai déjà défini la notion de référentiel. Je vous le rappelle d'une façon plus fonctionnelle.

Pour orthographier un accord duverbe, nous jouons avec :

. une règle : "l'accord du verbe se fait en personne et en nombre avec le sujet".

. une condition : notre petit élève doit reconnaître qu'un problème d'accord se pose à lui, il doit identifier le verbe et son sujet.

. un domaine d'application : le verbe de la phrase qui se présente.

La règle est généralement connue de l'enfant et ne pose pas de problème.
Le domaine d'application sera toujours le verbe de la phrase présentée, cela va de soi.

C'est donc très souvent la condition qui pose des difficultés.

La condition est chaque fois remplie quand la phrase est simple (quand le verbe suit immédiatement le sujet "comme dans le cahier de conjugaison").

Mais, dans la réalité, les phrases sont plus espiègles.
L'enfant ne voit plus qu'il est toujours aux prises avec un verbe et son sujet. Son référentiel est faux (cf.notion de référentiel).

2) créer un scénario.

Essayons donc d'imaginer un scénarion par lequel l'enfant sera amené à créer son propre "bon référentiel".

Messieurs Tonneau, Wéry et moi dégageons deux phrases :

1. amener l'enfant à remarquer l'existence d'une structure de phrase et diriger son intuition vers la règle à appliquer.
2. imprégner l'enfant de cette structure pour amener le réflexe vers la règle associée. :

   . d'abord en lui faisant reconnaître cette structure parmi

une liste de différents cas d'espèces.
. ensuite, en lui demandant de produire lui-même des structures similaires.

L'acquisition de la règle se fera tout au long de ces deux phases par le truchement d'abondantes variations sur les structures mises en jeu.

3) importance des variations.

a. définition et exemple.
b. pourquoi des variations
c. variations nécessaires;

-

a. définition et exemple.

d'abord, qu'est "une variation" ?
Je dirai qu'une variation est une modification de la présentation d'un exercice. Elle porte obligatoirement sur la forme, l'aspect extérieur, mais pas sur le fond, le raisonnement à suivre, la règle à appliquer.

Voici tout de suite un petit exemple :

Imaginons que l'objectif de notre exercice soit de familiariser l'enfant avec la notion de ce qu'on peut appeler un stock et de faire appliquer les règles de calcul + - x :

Voici le type d'exercices que nous utilisons (attention Messieurs les pédagogues, je ne cherche qu'à illustrer un concept, rien d'autre. Merci)

- un robinet coule dans un réservoir de 100 litres de capacité le débit est de 2 litres par 5 secondes. En combien de temps le réservoir sera-t-il plein ? (réponse en annexe).

une variation peut être : et si le réservoir est percé ... ?

ou plus astucieux (sur un autre thème) :

- Jean dépose 100 FB sur son livret d'épargne à 8% l'an. combien aura-t-il dans un an ? (réponse en annexe)

b. pourquoi des variations ?

Le motif est bien sûr pédagogique.
Tous les professeurs savent par expérience que chez les élèves, la difficulté résulte plus de l'application d'une règle apprise que de sa connaissance.

exemple : tous les enfants un peu avertis vous diront que "ON MET S au pluriel des noms communs".

Et pourtant, ils ne font pas souvent l'accord !

Admettons que je sois l'instituteur et que Thomas soit mon élève.
Animé d'une bonne intention, je lui soumets un exercice d'entrainement :

accorde les propositions :
. je veux les pomme_
. maman épluche deux poire_
. Christine aime les pêche_
. Oh! les beaux abricot_

Stupeur, Thomas fait zéro faute ! Je suis fier de moi, je jubile.

Le lendemain, au cours de la leçon de calcul, je lui dicte un petit énoncé : "papa achète au marché 3 kgs de pomme_ et 2 Kgs de poire_ ...
bien sûr, ni pomme, ni poire n'a pris une S.

Dans l'exercice que j'ai donné à Thomas, il n'avait en fait qu'à me répéter la règle et il la savait. Mais le lendemain son attention était concentrée sur les difficultés arithmétiques à venir. Il a oublié d'appliquer une règle simple qu'il connait parce que le contexte et la forme étaient différents.

Nos exercices viseront donc l'utilisation opportune des règles apprises.
Ce sont les variations qui permettront de quitter le contexte d'application immédiate. Ainsi égaré, l'enfant sera forcé de réfléchir devant chaque situation.

c. <u>variations nécessaires</u>.

En discutant avec les professeurs qui me conseillent, nous avons choisi que les variations nécessaires que doivent compter des exercices sur l'accord du verbe concernent la structure de la phrase, le verbe, le sujet.

. structure de la phrase.

pour réaliser des exercices sur les cas envisagés en page 6, il faudra :

- phrases avec sujet suivi du verbe
- phrases avec inversion
- phrases avec relative
- phrases avec écran entre le sujet et le verbe. cet écran pourra être un pronom ou un adverbe de manière.

. le verbe.

les verbes devront être variés en type et conjugaison.
type : nous avons des verbes en E (1er groupe classique)
IS (qui font issant au participe présent)
(N.B.) S (autres verbes courants)

conjugaison.
parmi les temps de l'indicatif présent, passé composé, imparfait, passé simple (très courant dans les rédactions d'élèves) futur simple, futur antérieur, futur (verbe aller + infinitif)

(NB) cette classification est tirée de E. De Meur-Muset - 6è année - éd.de boeck p. 128 - Tableau de Massarenti (psychopédagogie des moins doués)

. le sujet.

le sujet sera particulièrement varié

en type - le sujet sera un pronon personnel
un groupe nominal (nom, adjectif déterminant)
ou une suite de pronoms et de groupes (papa, maman, la bonne et moi)

en nombre - le sujet prendra la forme singulier ou pluriel
en genre - le sujet sera masculin, féminin ou impersonnel (on)
en personne - le sujet prendra les trois personnes existantes.

d. énoncé d'un exercice.

1) remarque préalable.

Cet exercice est en fait le seul que nous ayons imaginé. Il concerne les problèmes d'accord dans le cas d'inversion verbe-sujet.

La démarche pédagogique présentée doit encore être appliquée à la conception de scénarios sur les problèmes d'écran entre verbe et sujet, sujet qui, sujet tu, sujet on. Pour imaginer ces scénarios, je suisouvert à toutes les suggestions.

Dans ce qui suit, rien n'a ancore été prévu quant aux spécifications techniques (grilles d'écran, comment répondre aux questions, montrer, écrire, souligner etc ...)

2) scénario sur les inversions.

première phase : attirer l'attehtion de l'élève sur une structure particulière, ici l'inversion, diriger son intuition vers la règle.

Nous proposons à l'enfant trois phrases dans lesquelles il y a inversion.
Ces phrases sont variées en type de phrase (mais il faut une inversion), en type de verbe et de sujet.

Pour chacune de ces phrases, nous demandons à notre élève de la lire attentivement, de l'observer et de répondre à deux questions :

- quel est le sujet ?
- quel est le verbe conjugué ?

Le but des deux questions est de faire remarquer à l'enfant que le sujet est après le verbe et aussi que la règle d'accord sujet verbe est la même que dans le cahier de conjugaison.

S'il s'avère que l'enfant ne sait pas répondre, malgré plusieurs tentatives c'est qu'il n'est pas du niveau requis ou qu'il le fait exprès.
Un message à son instituteur mentionnant les faiblesses grammaticales est alors indispensable.

2ème phase. imprégner l'enfant de la structure inversion et engendrer le réflexe nécessaire.

a) pour cela, nous générons dix phrases variées selon leur type de structure, le verbe conjugé et le sujet. Il y aura au moins 4 inversions.

Nous ordonnons alors à l'élève :

. retrouve les phrases ayant la même caractéristique que les trois exemples du début.

L'enfant fera sans doute des erreurs. Pour chacune, nous allons lui faire comprendre ce qu'on attend précisément comme réponse.

. regarde les positions du verbe et du sujet dans les trois premiers exemples. Le sujet est-il avant ou après le verbe ?
. corrige ta faute.

b) écris trois phrases avec cette caractéristique.
Pour chaque phrase, deux traitements successifs :

. d'abord correction de la syntaxe et de la sémantique.
. ensuite correction des fautes d'accord du verbe et de respect de la caractéristique demandée.

En fait, concernant le second traitement, l'enfant peut avoir écrit :

. une phrase avec inversion et bien accordée.
. une phrase avec inversion et mal accordée
. une phrase sans inversion et bien accordée
. une phrase sans inversion et mal accordée.

Nous corrigeons d'abord l'oubli d'inversion et ensuite la faute d'accord.

Pour corriger l'oubli de l'inversion :

. on lui rappelle les 3 phrases du début et ce qu'il a dit sur la position du verbe et du sujet.

. on le questionne: dans ta phrase le verbe est-il avant ou après le sujet ?

. lui dire que dans les 3 phrases du début, le verbe est toujours avant le sujet et que l'on désire qu'il réécrive sa phrase en respectant cette propriété.

Pour corriger une faute d'accord :

quand je dis faute d'accord, ce n'est pas nécessairement vrai nous trouvons en fait :

. des fautes d'accord
. des terminaisons de verbe qui n'existent pas.

correction des fautes d'accord.

On lui donne alors à observer une quinzaine de phrases dont la structure d'inversion est la même que la sienne mais variées selon le verbe et le sujet.

L'enfant devra remarquer que le verbe change quand le sujet change.
Après cet étalage, on lui demande de corriger sa phrase.

correction de fautes de conjugaison (ou faute de frappe).

Selon nos hypothèses, nous sortons ici du cadre de l'accord du verbe, mais pour ne pas tourner court, nous proposons quand même un petit module de correction.

Voici ce module :

écris le verbe à l'infinitif .....
et classe le O E
O IS
O S

le conjugais-tu au présent O
passé composé O
imparfait O
passé simple O
passé antér. O
futur simple O
futur antér. O

Ensuite, lui présenter une quinzaine de groupes sujet/verbe

Chaque phrase sera du même groupe et au même temps de l'indicatif que le sien.

Les variations du sujet se feront en personne, nombre et genre (aumoins 4 sujets auront les mêmes propriétés que le sien).

Lui demander d'observer et de corriger sa faute. Il peut s'aider du tableau au mur.

Partout dans ce programme, les échecs répétés seront communiqués au maître d'école.

e. Contrainte sur le sens des phrases échangées entre l'enfant et son ordinateur.

1) contrainte pédagogique.

Deux solutions s'offraient à nous.
On exige que les phrases échangées aient un sens pour l'enfant, qu'elles viennent de la machine ou de l'enfant.

On ne l'exige pas.

Sachons que du point de vue pédagogique, les deux solutions sont défendables. Les uns vous diront que les questions d'orthographe dépendent essentiellement de la syntaxe dans les cas qui nous préoccupent. L'aspect sémantique pourra être délaissé. D'autres répondront qu'une phrase forme un tout en syntaxe et en sémantique. Séparer les deux conduirait à embrouiller l'esprit des élèves.
Des enfants de très bas niveau pourraient même être totalement perdus devant des phrases aussi originales que, par exemple :

. la radio promet mon boulanger au train.
. le vélo regarde le violent cinéma.

Monsieur Tonneau a ainsi particulièrement insisté pour que l'on retienne la première solution.

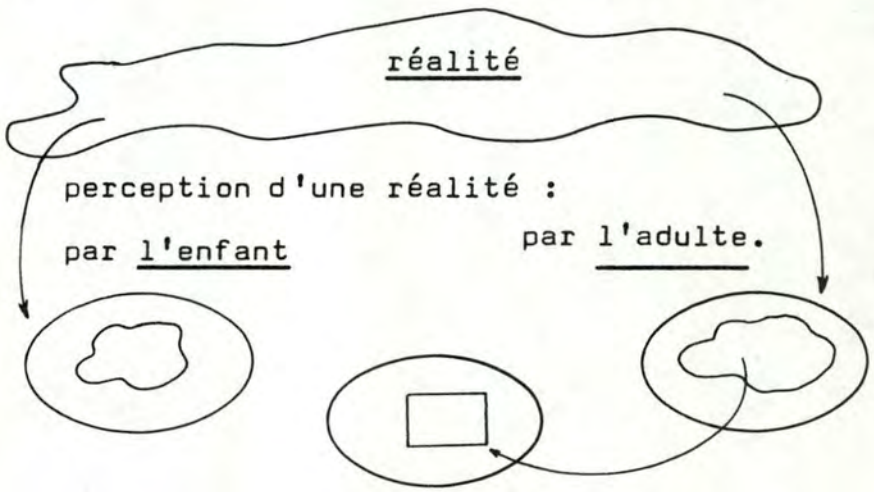
2) contrainte informatique.

Je n'ai pas pris position sur cet épineux problème; j'ai bien trop peu d'expérience dans ces questions.

Je choisis donc d'être le fournisseur et laisse à mes clients la responsabilité de leurs choix.

Avant de parler du contrat, je voudrais quand même préciser ce que peut signifier "sémantique" aujourd'hui en informatique dans le contexte de notre didacticiel.

Imaginez cet ensemble flou et mal connu qui est la réalité : et puis aussi trois entités qui nous concernent :



acquisition d'une réalité par l'ordinateur

L'enfant a une vision de la réalité qui lui est propre. L'adulte a lui aussi sa vision dela réalité, elle est plus scientifique, plus profonde.

L'ordinateur a la vision de la réalité que lui donne le programmeur, c'est une vue très partielle.

Pour décrire leur vision de la réalité, les entités disposent de leur propre langage.
Ce langage est composé de phrases. Dans le cadre de notre projet, nous dirons qu'une phrase est juste sémantiquement pour une entité si ce que cette phrase décrit appartient à la vision de la réalité de cette entité.

Observons donc qu'une phrase peut être sémantiquement juste pour celui qui la dit et pas pour celui qui la reçoit.

."Tous les éléphants sont roses". Cette phrase aura généralement un sens pour un enfant ou un ivrogne, mais amusera un zoologiste.

."E = MC2" risque de ne pas signifier grand chose chez un enfant de 10 ans, si ce n'est une citation historique.

."je veux manger des frites" sera compris par quiconque a vécu un mois en Belgique.

."le pré broute la vache ", je ne connais personne qui puisse lui donner un sens.

Ceci nous amène à la communication ordinateur/enfant.

Une phrase proposée par l'enfant à l'ordinateur sera acceptée comme juste sémantiquement par l'ordinateur tant quela réalité que recouvre la phrase appartient aussi à la réalité de l'ordinateur.

Vice versa, une phrase proposée automatiquement à l'élève sera comprise si la réalité que recouvre la phrase appartient aussi à la réalité de l'enfant. En pratique, c'est difficile, c'est de l'habileté du programmeur à cerner la réalité de l'enfant que dépend le succès de l'entreprise.

Je serai donc très prudent quant au résultat que nous pouvons obtenir.

Pour les échanges de l'ordinateur vers l'enfant :

. j'avance que 90 % de mes phrases sont sémantiquement justes pour un enfant.

. je ne garantis pas du tout qu'elles sonnent bien à l'oreille

. je vous certifie que 90 % de mes phrases vous feront sourire.

Pour les échanges de l'enfant vers la machine :

. je ne peux pas vous jurer que mon programme ne refusera pas parfois une phrase acceptable dans l'esprit d'un enfant.

. si l'enfant écrit volontairement des bêtises, je ne garantis pas que mon ordinateur les rejette toutes.

Il faudra aussi accepter que le vocabulaire qui composera les phrases soit très réduit.

B. Elaborer une solution informatique.

1. introduction.
2. définition des données - classes sémantiques
   - choix de données
3. définition des grands traitements
4. structure de la base de données
5. les deux applications existantes.

-

1. introduction.

Peut-être avez-vous frémi en imaginant les outils informatiques nécessaires à la réalisation du projet global. Il est vrai que leur construction n'est pas triviale. Il faudra inventer:

a. un programme capable d'illustrer des points précis de la grammaire française par des exemples qui seront des phrases cohérentes.

Il devra pour cela pouvoir construire desphrases selon toutes les variations possibles de types de phrases, de verbes, de sujets (cf le paragraphe variations nécessaires, p.) et en présentant un bel emballage : adjectifs, adverbes, compléments divers.

Il faudra s'arranger pour que les phrases soient sémantiquement acceptables pour un enfant de dix à douze ans.

Dans l'éventualité d'extensions futures, il serait intéressant que ces phrases soient aisément manipulables; (je pense à des pronominalisations, des inversions ...)

b. un programme qui décodera des phrases proposées par l'enfant, régulièrement truffées des erreurs les plus grossières.

Je distinguerai comme types de fautes :

. les fautes de syntaxe grammaticale, l'ordre des mots ne correspond pas à la syntaxe du français.
. les fautes d'accord qui ne sont pas liées au problème qui nous concerne, accord de noms, d'adjectifs ...
. les fautes d'accord du verbe avec son sujet et de conjugaison.
. les fautes d'orthographes d'usage et les fautes de frappe.
. les malentendus sémantiques.

Pour toutes les fautes et malentendus, à l'exception des fautes d'accord, sujets et verbes, le programme devra prévoir un traitement standart propre à signaler l'obstacle à l'enfant et à corriger ou faire corrigrr la faute en un minimum de temps. La raison est que nous souhaitons atteindre au plus vite l'objectif "accord du verbe". Pour l'éxception, les fautes d'accord sujet et verbe, le programme se contente de les signaler au programme principal en lui donnant l'environnement de la faute, la phrase en paramètre.

c. un programme qui puisse enregistrer l'évolution de l'élève dans le temps.

Ce programme devra fournir pour chaque écolier une série de statistiques concernant les types de problèmes sur lesquels il butte et aussi sur son évolution au cours du temps.

Ces statistiques seront présentées pour qu'un programme puisse les lire et orchestrer les types d'exercices qu'il présentera ultérieurement à l'enfant. Il s'agira aussi que l'instituteur puisse lire ces statistiques, éventuellement les modifie, et juger, grâce à ces statistiques et à son expérience de l'élève, de la validité de ce projet d'EAO.

Les points a et b. qui traitent de manipulations de phrases impliquent le choix de structures de phrases et de vocabulaire.

Pour des raisons de simplicité, les générateurs et décodeurs de phrases fonctionneront sur un vocabulaire et une structure de phrase strictement identique.

2. définition des données.

a. préambule et notion de classe sémantique.
b. choix du vocabulaire
c. choix des compléments à une structure de phrases.

Contrairement à ce qui se passe souvent dans la discipline informatique, je dois non seulement décider d'une structure de données qui permette un traitement aisé, mais aussi décider de la valeur de ces données - choix du verbe et des compléments aux structures de phrases nécessaires.

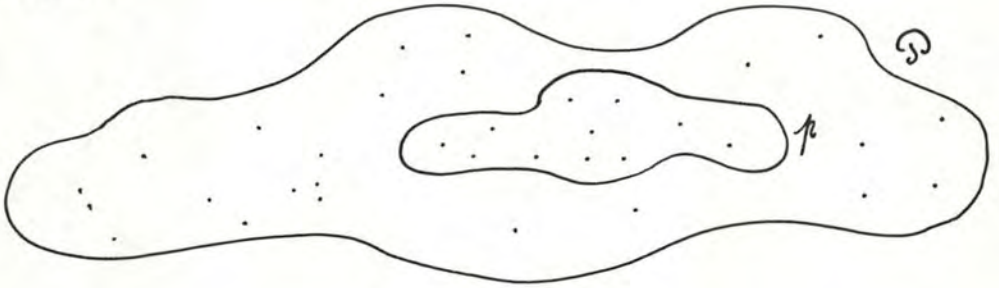
C'est une des originalités qui contribuent à rendre mon mémoire amusant.

a. préambule et notion de classe sémantique.

Il me serait très difficile de vous expliquer indépendamment comment j'ai choisi mon vocabulaire et mes garnitures de phrases.

C'est que pour résoudre ce problème, il m'a fallu réfléchir à une idée de solution concernant l'aspect contrôle sémantique. Monsieur Cherton et moi en avons trouvé une, elle repose sur la notion de découpe en classes.

Soit $\mathcal{P}$ l'ensemble des phrases sur le vocabulaire de notre langage.



En observant cet ensemble, on parvient à isoler les phrases p du type

"quelqu'un ou quelque chose exécute une action qui porte sur "quelque chose dans un environnement de lieu et/ ou de temps".

ex. Nicolas mange un abricot.
Christine frappe David dans la cuisine.

L'idée est que pour certains types d'actions, simples et non ambigus, il est possible de trouver des propriétés des êtres ou des choses qui exécutent ces actions, des êtres ou des choses sur qui portent ces actions et l'espace dans lequel ces actions peuvent se dérouler.

Par exemple, l'action de manger.

. quelles sont les propriétés de tout ce qui peut exécuter l'action de manger ?

Je trouve que tout ce qui est être vivant peut manger, c'est à dire Nicolas, mon chat, un piranha etc ...

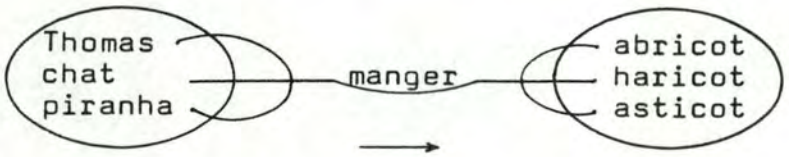
. sur quoi peut porter l'action de manger ?

sur tout ce qui est comestible d'une façon très large, c'est-à dire : un abricot, des haricots, des asticots etc ...

Ayant énoncé la liste des propriétés dont nous avons besoin pour décrire l'environnement des types d'actions retenues, il reste à puiser dans le vocabulaire français des mots qui vérifient ces propriétés et à remplir ce que j'appellerai des classes sémantiques.

A partir d'une action et des classes sémantiques de ceux qui exécutent et de ceux sur qui porte l'action, je peux déjà combiner beaucoup de phrases sensées différentes.

petit exemple :



neuf phrases différentes dans ce cas simple.

b. choix du vocabulaire.

Il s'agira de sélectionner les noms communs, les noms propres les verbes, les adjectifs et les adverbes qui formeront mes phrases.

Mon choix respectera deux contraintes :

. le vocabulaire sera connu de l'écolier.

pour m'aider, Monsieur Tonneau m'a donné une liste de 1.800 mots que tout élève devrait connaître en 6è année primaire (vous trouverez cette liste en annexe). Je me suis basé sur cette liste.

. ce vocabulaire sera facile à trier en classes sémantiques.

Il m'appartiendra de me montrer suffisamment judicieux pour que ça marche

Ma méthode sera basée sur la notion de découpe en classes que je vous ai décrites. Je choisis de partir du centre de la phrase, là où se situe l'action, donc du verbe.

Choix des verbes.

les contraintes sur les verbes sont d'utiliser des verbes courants, variés en groupe de conjugaisons E/IS/S (voir annexe). D'un point de vue sémantique, il importe qu'ils ne soient pas ambigus.

En parcourant la liste de Monsieur Tonneau, j'ai trié les différents groupes de verbes. J'en ai dénombré 275 dont 70 % de "E", 5 % de "IS", 25 % de "S".
J'ai décidé de suivre cette tendance.

Dans chaque groupe, j'ai choisi les verbes qui m'ont paru les moins ambigus (C'est très arbitraire, bien sûr !) et les moins restrictifs.

J'ai évité tous les verbes d'état et les auxiliaires être et aller. J'ai quand même choisi d'accepter le verbe être si c'est l'enfant qui veut l'employer. C'est quand même un des verbes les plus courants, mais je ne l'utiliserai pas en génération.

Je me suis aussi arrangé pour couvrir plusieurs thèmes et pour éviter les synonymes du style manger, souper, déjeuner, goûter.

Voici quelques exemples :

Je préfère le verbe laver au verbe raser, car on lave beaucoup plus de choses qu'on n'en rase (raser est plus restrictif)

Je préfère étudier au verbe travailler. Etudier est plus restrictif que travailler, mais travailler peut être ambigu.

En effet, David ne travaille pas, mais aussi, le bois travaille beaucoup.

Je rejette le verbe couler parce qu'il est trop ambigu.
En effet, les pirates ont coulé le bateau et le sang a coulé aussi.

Des verbes comme cueillir, attraper, parler ... se prêtent très bien à l'emploi que je leur ai destiné. Les sujets de ces verbes auront toujours la même propriété (être vivant animé).

J'ai finalement retenu 21 verbes en ER, 7 verbes en IS 16 verbes en S soit 44 verbes distrubués par groupe, plus ou moins comme dans la langue parlée.

Remarquons que sur la totalité du vocabulaire de 250 mots, la distribution des verbes reflète assez bien la réalité.

Choix de noms communs, d'adjectifs et d'adverbes.

En sélectionnant des verbes, je me suis intuitivement constitué une vingtaine de types de classes. Il me restait à les remplir de vocabulaire. J'ai choisi des mots extrèmement courants au gré de mon imagination.

J'ai accompagné ces noms communs de 25 adjectifs qualificatifs très répandus.

Quant aux adverbes, ils dépendant des actions qu'ils précisent. Je les ai choisis très "passe-partout". J'en ai retenu 26.

Tout le vocabulaire choisi se trouve énuméré en annexe.

c. Choix des structures de phrases.

1. motivation de mon choix.
2. description de la syntaxe.

1) motivation de mon choix.

Je vais vous avouer que j'étais très ambitieux au départ, trop même. Je n'étais franchement pas raisonnable. Je voyais des subordonnées, des circonstances de temps, de lieu, de manière, de but, de cause ... que j'étais sot !

Une étude plus approfondie m'a vite contraint à limiter très sérieusement tout cela.

En fait, il me fallait des règles plus simples, peu nombreuses et offrant un maximum de possibilités.
Je devais faire la chasse aux exceptions. Ceci pour deux raisons :

- d'abord il faut aider l'utilisateur. Il est inutile d'essayer de faire retenir à l'élève des règles de syntaxe, même très simples. Il les mélangerait et les oublierait. L'enfant apprendra ce qu'il peut et ce qu'il ne peut pas écrire en observant ce que la machine produit.

  J'espère que l'enfant remarquera les limitations de la machine et découvrira les règles.

- ensuite, je dois penser à moi. Plus les règles sont compliquées, plus il me sera difficile de les programmer et plus je ferai d'erreurs.

J'ai donc opté de travailler avec principalement trois types d'objets.

. des verbes, des adverbes, des groupes noms.

Vous connaissez les verbes et les adverbes.

Je vais vous présenter le groupe noms de mon projet, beaucoup plus pauvre que ceui que vous connaissez.

Ce groupe noms est composé d'un nom commun et de quelques fioritures qui sont adjectifs qualificatifs, démonstratifs, possessifs, numéraux, définis, indéfinis.

par exemple : un canard bleu
mes trois petites amies
le méchant gamin.

Il pourra prendre la fonction sujet, complément d'objet direct complément d'objet indirect; ces deux dernières fonctions seront d'ailleurs presque toujours des groupes noms (presque parce qu'il existe aussi des pronoms

2) Description de la syntaxe.

a. objectif.
b. problèmes rencontrés
c. syntaxe en annexe. p.A.I.1

a. quel est l'objectif de cette description ?

Il est essentiel d'informer les gens, informaticiens, pédagogues qui liront ce mémoire. Mon souci est qu'ils aient une bonne idée des types de phrases étudiées et puissent critiquer mon choix. J'aimerais aussi documenter tout étudiant pour qui se poserait ce genre de problème, le choix de structure de phrases.

L'objectif n'est pas de me faciliter la vie pour l'implémantation. Je n'ai pas travaillé en équipe et n'ai donc pas eu de problème de communication. Comme programmeur solo, je me suis contenté de notes aide-mémoire personnelles. J'avais la syntaxe en tête.

b. problèmes rencontrés

Cette syntaxe ne traite pas des règles d'accord. Elle présente plutôt l'ordonnancement des mots de différentes natures pour constituer une phrase.

Comment vous la présenter ?
J'ai deux problèmes : un petit et un gros.

Le petit : sous quelle forme puis-je vous présenter cette syntaxe ?

- sous forme de dessin.
  Je pense aux diagrammes qui décrivent la syntaxe du langage Pascal dans la majorité des ouvrages sur ce sujet.

- sous forme de texte
  Je pense à une description en B.N.F.

Pour des raisons de lisibilité, je choisis la première méthode. Attention, je la trouve peut-être plus lisible parce que j'y suis plus accoutumé ! C'est très arbitraire, au fond.

Le gros problème, maintenant.

Vu de loin, je croyais fermement qu'il était très simple de formaliser la syntaxe française si je m'en tenais à un petit sous-ensemble bien choisi. J'ai vite déchanté.

Il y a des tonnes d'exceptions.

Ex. Voici une phrase dont le complément d'objet indirect est à la 3è personne.

. j'écarte le gros monsieur qui parle à Benoit.

La règle de syntaxe semble évidente : verbe/à/nom propre ou pronom.

Essayons avec un COI à la Ière personne, nous aurions :

. j'écoute le gros monsieur qui parle à moi.

ça y en a faire petit nègre, n'est ce pas ?
en bon français on dira plutôt :

. j'écoute le gros monsieur qui me parle.

Il n'y a plus de préposition à et le pronom est propulsé devant le verbe.

Pas simple, n'est ce pas ?

Ex. Une règle générale de l'inversion est de placer un trait d'union entre le verbe et le sujet.

Voyons ...

. mangeait-il ?
C'est la règle générale. Il n'y a rien à dire.

. mange-t-il ?
Il y a introduction d'un symbole parce que la conjugaison du verbe change.

Mon gros problème est le suivant : Si des langages conçus pour l'informatique comme les langages de programmation, les langages querry et autres langages de description de données se prêtent très bien à une description formelle, ce n'est plus le cas des langages naturels, le français en tous cas.

Je n'entrevois que deux possibilités :

a. je tente malgré tout une description complète. Je risque alors de vous donner 25 pages de diagrammes et d'erreurs subtiles.

b. je donne une description de la règle générale et je l'accompagne d'un commentaire sur les exceptions. Je spécule ici sur le fait que je décris quelque chose que vous connaissez, votre langage.

Je retiendrai cette seconde possibilité.

c. La description de la syntaxe se trouve en annexe : A.I.1

3. définition des grands traitements.

Quatre programmes sont particulièrement intéressants dans mon projet.

a. un générateur de phrase.

Oh !, ce n'est pas générer des phrases qui est difficile. Construire une phrase n'est en fait que suivre un chemin dans les diagrammes syntaxiques en allant chercher des mots dans une base de données.
Ce qui est dur, c'est de bien organiser cette base quant à sa forme (rapidité d'accès, économie de place mémoire) et son contenu (choix des mots, règles d'accord des noms, d'adjectifs et verbes, règles de sémantique).
Le générateur nous permettra surtout de visualiser la qualité de la base de données.

La base de données existe, je vous en parlerai à la page suivante.

Le générateur existe également. Vous trouverez en annexe un large échantillon de ce qu'il produit, le texte du programme et sa documentation.

b. correction automatique des fautes d'orthographe d'usage et des fautes de frappe.

C'est un programme que l'on utilisera quand on décodera les phrases de notre élève.

Le but du projet est l'accord du verbe. Pour aller droit à ce but, nous aimerions perdre un minimum de temps dans tous les autres problèmes. Corriger automatiquement les fautes d'orthographe d'usage et les fautes de frappe fera gagner beaucoup de temps.

Nous sommes deux étudiants intéressé par un tel programme, Jacques Kinet qui l'emploiera dans le cadre de son projet EAO en langue espagnole et moi. Nous avons donc collaboré tous les deux à sa réalisation.

Nous sommes arrivés à une solution très acceptable.
Vous trouverez en annexe le texte du programme et sa documentation.

c. le décodeur de phrases.

La base de données et la correction automatique existent. L'interprêteur syntaxique et le décodeur sémantique n'existent pas encore.

J'entrevois cependant les quelques problèmes qui surviendront.

. diminution des performances de la correction automatique à cause de la prise en charge des accords (un verbe peut avoir jusuqu'à cinquante orthographes différentes).

. correction automatique si l'enfant écrit une phrase selon une structure qui n'est pas reprise dans mes spécifications.

. la taille du programme se rapprochera dangereusement du plafond imposé par la machine.

d. suivi de l'enfant.

Ce programme tiendra des statistiques sur l'évolution de l'élève et devra se baser sur ces statistiques pour proposer à chaque élève ce qui lui convient le mieux au point de vue degré de difficulté et type d'exercice.

Il n'existe pas, mais je devine déjà que lavalidité de ces statistiques sera surtout fonction de la qualité des scénarios que l'on inventera.

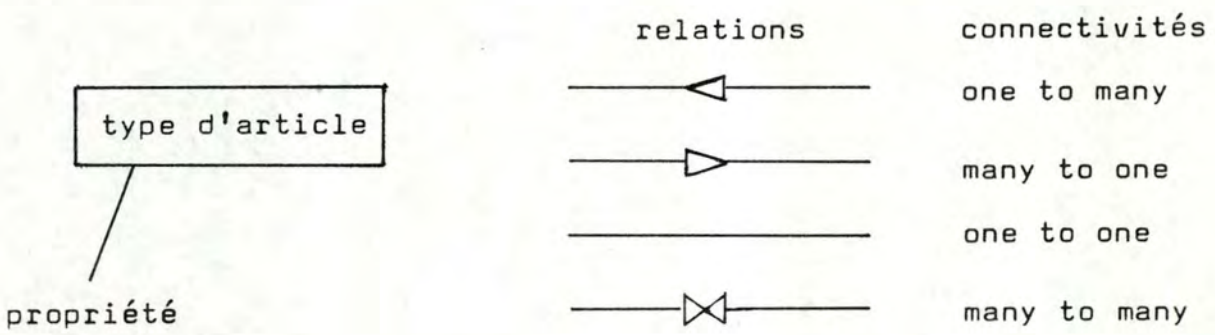
4. Structure de la base de données.

Je vous rappelle encore que cette base de données est commune au générateur de phrases et au décodeur de phrases.

Elle contient l'information nécessaire à la construction de phrases syntaxiquement correctes, sémantiquement acceptables et au décodage syntaxique et sémantique des phrases de nos petits élèves.
L'information est donc structurée en vue de ces traitements.

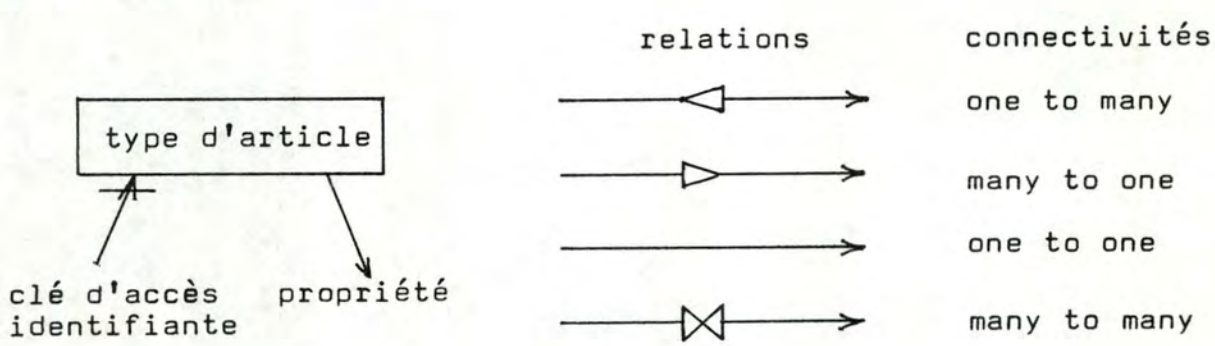
Je vais d'abord vous présenter l'organisation des données :
la syntaxe et puis la sémantique.
Je m'aiderai d'un modèle binaire pour communiquer.

bref rappel



Pour terminer, je vous presenterai un modèle d'accès de l'ensemble de la base de données.

bref rappel

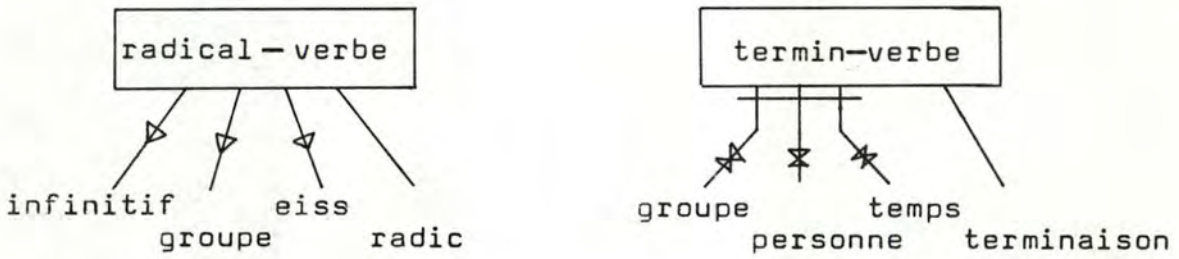


a. Aspect syntaxique.

. le verbe.

La structure d'information concernant la syntaxe des verbes doit nous permettre de conjuger chacun des 44 verbes à l'un des huit temps de l'indicatif et ce pour n'importe quelle personne. Il faut aussi pouvoir distinguer ces 44 verbes en 3 groupes E/IS/S.

Sous-Schéma.



radical-verbe.

le radical-verbe est la partie invariable d'un verbe conjugé. Dans mon vocabulaire, tous les verbes ont des radicaux différents.

Infinitif : orthographe de la partie variable à l'infinitif.

Radic : orthographe de la partie invariable d'un verbe conjugué, elle contient au moins un caractère.

EISS : E/IS/S, détermine l'appartenance du verbe à un de ces trois groupes.

Groupe : La notion de groupe est beaucoup plus profonde que la notion EISS. Si l'on consulte le guide Bescherelle "l'art de conjuguer", on constate qu'il existe 82 paradignes permettant de conjuguer tous les verbes. Ce que j'appelle groupe est un de ces paradignes.

Termin-verbe.

partie variable d'un verbe conjugué.

groupe : cf radical-verbe.

personne : je, tu, il, nous, vous, ils

temps : présent, imparfait, passé simple, futur simple, participe passé.

terminaison : orthographe de la partie variable d'un verbe conjugué

Ne vous affolez pas, je n'ai pas oublié d'information dans ce sous-schéma. Pour construire un temps composé, il suffit de composer le participe passé du verbe à conjuguer et l'auxiliaire être ou avoir à un temps simple adéquat.

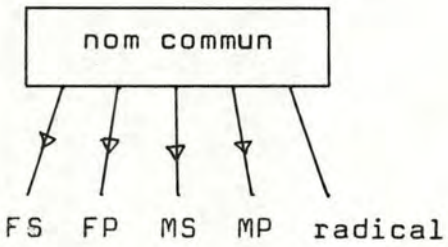
exemple :

Passé composé = présent de l'auxiliaire + participe passé
plus que parfait = imparfait de l'auxiliaire + participe passé
passé antérieur = passé simple de l'auxiliaire + participe passé
futur antérieur = futur simple de l'auxiliaire + participe passé

. nom commun.

La structure d'information concernant la syntaxe des noms communs doit nous permettre d'accorder ces noms communs au féminin, au masculin, au singulier et au pluriel pour autant qu'ils acceptent ces différentes formes.

sous-schéma.



radical : orthographe de la partie invariable d'un nom commun, elle contient au moins un caractère.

FS : orthographe de la terminaison du nom commun au féminin singulier. Si ce nom commun n'admet pas de féminin singulier, le champ est vide.

FP : orthographe de la terminaison du nom commun au féminin pluriel. Si ce nom commun n'admet pas de féminin pluriel le champ est vide.

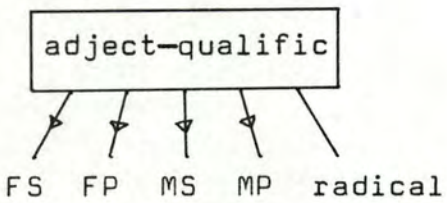
MS : orthographe de la terminaison du nom commun au masculin singulier. Si ce nom commun n'admet pas le masculier singulier, le champ est vide.

MP : orthographe de laterminaison du nom commun au masculin pluriel. Si ce nom commun n'admet pas de masculin pluriel, le champ est vide.

. adjectif qualificatif.

La structure d'information concernant la syntaxe des adjectifs qualificatifs doit nous permettre d'accorder ces adjectifs au féminin, au masculin, au singulier et au pluriel, pour autant qu'ils acceptent ces différentes formes.

sous-schéma.



radical : orthographe de la partie invariable de l'adjectif, elle contient au moins un caractère.

FS : orthographe de la terminaison de l'adjectif au féminin singulier. Si cet adjectif n'admet pas de féminin singulier, le champ est vide.

FP : orthographe de la terminaison de l'adjectif au féminin pluriel. Si cet adjectif n'admet pas de féminin pluriel, le champ est vide.

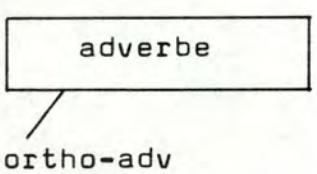
MS : orthographe de la terminaison de l'adjectif au masculin singulier. Si cet adjectif n'admet pas de masculin singulier, le champ est vide.

MP : orthographe de la terminaison de l'adjectif au masculin pluriel. Si cet adjectif n'admet pas de masculin pluriel, le champ est vide.

. adverbe.

L'information de la base de données doit nous permettre décrire un adverbe sans faute.

sous-schéma.

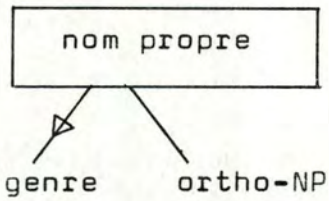


ortho-adv : orthographe de l'adverbe.

. nom propre.

L'information de la base de données concernant les noms propres doit renseigner l'orthographe du nom propre et son genre féminin du masculin.

sous-schéma.



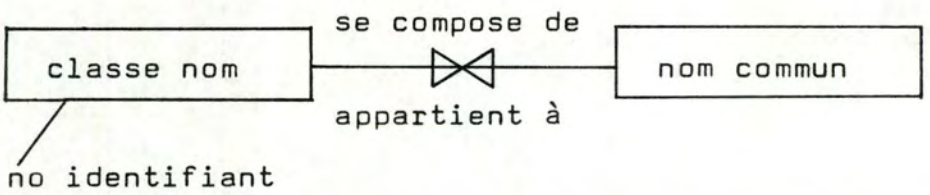
ortho-NP : orthographe du nom propre.
genre : féminin ou masculin.

b. L'aspect sémantique.

La structure de l'information relative à l'aspect sémantique doit permettre la répartition du vocabulaire en classes et l'utilisation de ces classes pour contrôler les associations verbes/sujet, compléments, adverbe et noms communs/adjectif dans les sens recevoi et générer.

. classes noms.

sous-schéma.



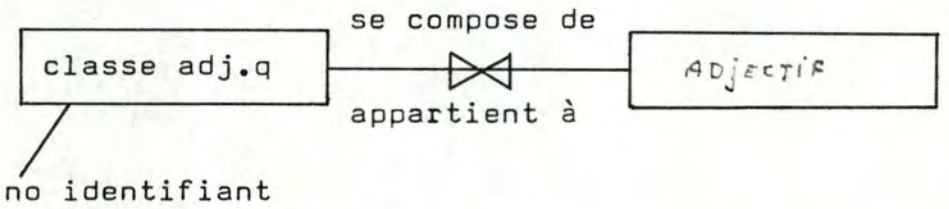
une exception : tous les noms propres forment une classe également.

Remarque : Dans mon programme, la connectivité de la relation est one to many dans le sens "se compose de". Il serait pourtant très vraisemblable qu'un mot un peu ambigu appartienne à plusieurs classes. La base de données l'admettrait.

classe nom : représente un ensemble de noms communs ayant une propriété commune telle que si un verbe accepte un des noms pour exercer une fonction quelconque, alors il accepte tous les autres pour la même fonction.

. classe adjectif.

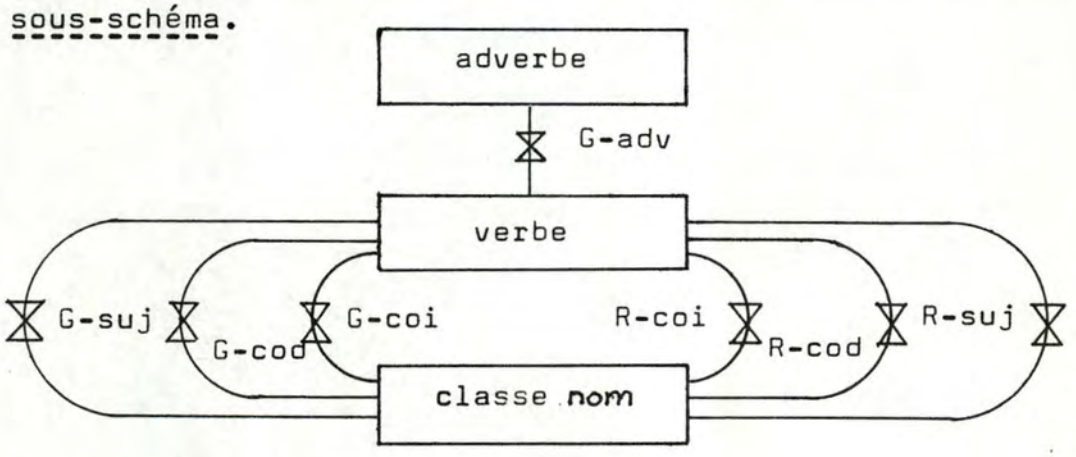
sous-schéma.



classe adj.q.: représente un ensemble d'adjectifs qualificatifs ayant une propriété commune telle que si un nom commun accepte un de de ces adjectifs, il les accepte tous.

. Relations entre les verbes , les classes noms et les adverbes

sous-schéma.



Nous avons sept relations many to many.

dans le sens générer

G-suj : une occurence de la relation G-suj a lieu lorsque une classe de mots est acceptée par un verbe pour que chacun de ses composants puisse jouer la fonction sujet de ce verbe dans le sens générer.

G-cod : une occurence de la relation G-cod a lieu lorsque une classe de mots est acceptée par un verbe pour que chacun de ses composants puisse jouer la fonction complément d'objet direct de ce verbe dans le sens générer.

G-coi : une occurence de la relation G-coi a lieu lorsque une classe de mots est acceptée par un verbe pour que chacun de ses composants puisse jouer la fonction complément d'objet indirect de ce verbe dans le sens générer.

G-adv : une occurence de la relation G-adv a lieu lorsque un adverbe est accepté par un verbe dans le sens générer.

dans le sens recevoir.

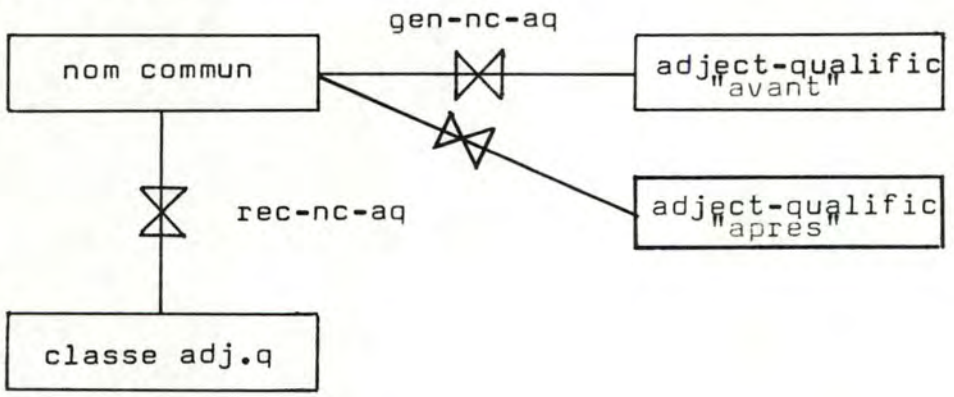
R-suj. : une occurence de la relation R-suj. a lieu lorsque une classe de mots est acceptée par unverbe pour que chacun de ses composants puisse jouer la fonction sujet de ce verbe dans le sens recevoir.

R-cod : une occurence dela relation R-cod a lieu lorsque une classe de mots est acceptée par un verbe pour que chacun de ses composants puisse jouer la fonction COD de ce verbe dans le sens recevoir.

R-coi : une occucence de la relation R-coi a lieu lorsque une classe de mots est acceptée par un verbe pour que chacun de ses composants puisse jouer la fonction COI de ce verbe dans le sens recevoir.

. relation entre les noms communs et les adjectifs.

sous-schéma.



Contrainte.

Par rapport à un nom commun, les adjectifs peuvent se diviser en deux groupes. Il y a les adjectifs qui sonnent mieux à l'oreille quand ils sont placés avant le nom et ceux que l'on préfère après le nom.

La contrainte est que chaque nom commun sera accompagné d'adjectifs "avant" et d'adjectifs "après" dans le sens générer.

gen-nc-aq

une occurence de cette relation a lieu quand un adjectif est accepté par un nom commun dans le sens générer.

rec-nc-aq

une occurence de cette relation a lieu quand une classe d'adjectifs est acceptée par un nom commun pour que chacun de ses composant joue la fonction adjectif de ce nom.

Remarque.

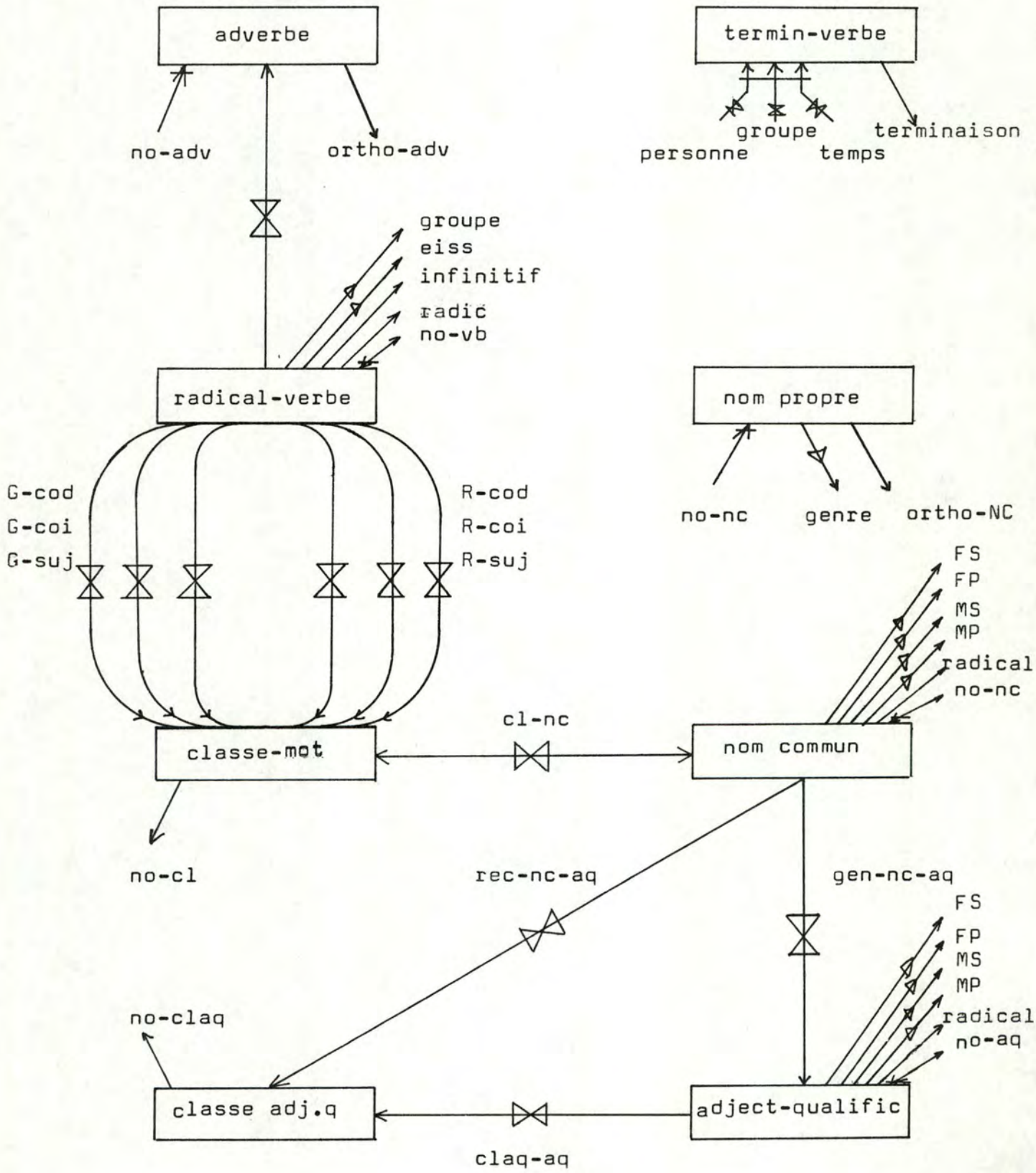
Vous avez remarqué que dans le sens générer j'associe directement un nom avec des adjectifs alors que dans le sens recevoir j'associe le nom avec des classes d'adjectifs groupés par propriété.

C'est parce que lorsque l'ordinateur construit une phrase il faut qu'elle soit comprise par l'élève.
Lorsque c'est l'enfant qui en proposera une, on lui fera confiance. On vérifiera seulement qu'il ne dépasse pas certaines limites très subjectives.

Si dans le sens générer j'ai choisi d'affecter à chaque nom commun une liste d'adjectifs plutôt que des classes, c'est parce que mon vocabulaire d'adjectifs est trop réduit (moins de trente) pour faire des classes

* Ne cherchez pas les pronoms et les conjonctions dans cette base de données, ils n'y sont pas. J'ai choisi de les générer par programme pour des questions de facilité.

c.intégration des sous schémas binaires en un modèle d'accès.



Je présenterai seulement les accès nécessaires au traitement sémantiqu

Ce traitement comporte deux applications.
L'une dans le sens où l'on génère des phrases, l'autre quand nous vérifions la sémantique de phrases proposées par les petits élèves.

1) dans le sens générer.

Pour déterminer les accès nécessaires, il m'a bien sûr fallu anticiper la technique de construction de phrases. J'emploierai pour générer des phrases le principe qui a déjà guidé le choix du vocabulaire. A partir d'un centre, le verbe, je choisis les noms communs et les adverbes qui peuvent l'entourer, à partir d'un nom commun, je choisis les adjectifs qui le qualifient.

Cela signifie que :

. d'un verbe, jedois pouvoir accéder à ces classes de mots pour chacune des trois fonctions (sujet, COD, COI) et à ses adverbes (chemins G-suj, G-cod, G-coi, G-adv).

. d'une classe de mots, je devrai accéder à l'ensemble de ces mots (chemin cl-nc)

. d'un nom commun, j'accèderai à l'ensemble des adjectifs qui le précèdent et qui le suivent (chemin gen-nc-aq).

2) dans le sens recevoir.

J'anticipe également que le contrôle sémantique suivra le même principe (du centre vers l'extérieur) que la génération.

Attention, rien ne dit que c'est le meilleur système. Je n'ai pas encore écrit ce traitement. Peut-être eut-il mieux valu un contrôle de l'extérieur vers le centre.

(ex. pour un nom commun, sujet, quel estl'ensemble des verbes possibles ?) Mais il fallait faire un choix - j'ai suivi mon intuition.

Cela signifie que :

. d'un verbe, je dois pouvoir accéder à ses classes de mots pour chacune des trois fonctions (sujet, COD, COI) (chemins R-suj. R- cod R- coi).

. d'un nom commun, il faut accéder à sa classe pour savoir si elle appartient à l'ensemble des classes permises par le verbe pour la fonction considérée (chemin cl-nc).

- d'un nom commun, il faut accéder à ses classes adjectifs (chemin rec-nc-aq)
- d'un adjectif, je dois accéder à l'ensemble de ses classes (chemin claq-aq)

5. Les deux applications existantes.

a. la correction automatique des fautes d'orthographe, d'usage et des fautes de frappe.

b. la génération des phrases.

a. la correction automatique des fautes d'orthographe, d'usage et des fautes de frappe.

1. utilité et exigences.
2. analyse

1. utilité et exigence.

Quant quelqu'un, "l'homme de la rue", écrit un texte ou une phrase, il fait des fautes : par distraction, parce que de bonne foi, il confond deux orthographes, parce qu'il se fie à son oreille. C'est ainsi que je fais énormément de fautes; et si j'en retrouve en me relisant, il en reste encore. Bien souvent, chez l'enfant, c'est encore plus marqué. Si en plus on frappe à la machune, il y a les fautes de frappe. Le doigt glisse sur une touche et hop! on est vu.

Mais voilà, notre projet devra décoder de telles phrases entachées d'erreurs. Or, à priori, un ordinateur ne connait que lesmots qu'on lui a donnés et ignore les autres. Un mot fautif pour la machine, c'est un *autre* mot.

Si j'ai appris à l'ordinateur "Alligator" et que je lui écris "Aligator" ou "alliator", il ne connaît pas.

Comme le projet ne peut pas se planter à chaque faute, il faut réagir. Je ne vois que deux solutions :

. à chaque faute, on demande à l'enfant d'écrire son mot autremen L'inconvénient est que la majeure partie du temps consacré à l'exercice risque de se disperser en corrections et corrections de corrections.

. une meilleure méthode serait alors d'apprendre à l'ordinateur à corriger lui-même ces types d'erreurs. Elle ne serait meilleure bien entendu qu'à deux conditions :
- le temps de réponse est suffisamment court, une moyenne de 5 secondes,
- le mot qu'il propose correspond assez souvent à ce que l'enfant voulait écrire, moins de 5 % d'erreurs.

2. analyse.

a. idées de solution.
b. bibliographie.
c. solution retenue.

Avant de vous exposer quoi que ce soit, je dois vous dire que l'analyse de cette application résulte essentiellement d'une collaboration. Jacques Kinet réalise lui aussi un mémoire en E.A.O. et dans son projet il demande aussi à ses élèves de construire des phrases, mais en espagnol.

Comme les problèmes d'orthographe d'usage et de fautes de frappe nous étaient communs, nous avons fait équipe pour les résoudre.

a. idées de solution.

L'ordinateur dispose d'une liste de vocabulaire. Pour lui, chaque mot est une chaine de caractères alphabétiques contigus. Ce qu'on appelle souvent un STRING.
Face à cetteliste, un mot erroné, c'est à dire une suite de caractères qu ne connaît pas notre machine.

Corriger l'erreur consistera à rechercher dans la liste le mot que probablement l'enfant a voulu écrire. Mais quel est ce mot ?

C'est ici que deux pistes s'offrent à nous.

. la ressemblance phonétique, l'enfant y fait souvent appel quand il ignore l'orthographe précise.
ex. un diament pour un diamant.
un dinosore pour un dinosaure.

. les fautes plus classiques (ajout, doublement, oubli d'une lettre) et fautes de frappe (inversion, oubli, ajout, remplacement d'une lettre, caractère illicite)

ex : une miete de pain, pour une miette de pain.
inevrsion pour inversion

Ces deux pistes nous semblent malheureusement difficiles à intégrer dans un sytème performant.

Ayant cerné le problème, il est temps pour nous de fouiller un peu livres et revues;

b. bibliographie.

on ne peut pas ire que nos recherches aient été très fructueuses. du côté de la piste "phonétique", nous avons fait "chou blanc". Mais en suivant l'autre voie, nous avons entr'autres, découvert deux articles assez alléchants.

Le premier est de Robert A. Wagner - Vanderbilt University, Naschville, Tenessee
et Michael J. Fischer - Massachusetts Institute of Technology.
the string to string correction problem.

Le second sera de Ronald W. Cornew - Massachusetts Institute of technology.
A statistical method of spelling correction.

-

1. The string to string correction problem

Les auteurs nous présentent une technique propre à calculer une "distance" entre deux strings. Ils se basent sur le nombre d'opérations élémentaires nécessaires pour rendre un des strings équivalent à l'autre.

ils emploient trois opérations élémentaires :

. remplacement d'une lettre par une autre dans le string.
. suppression d'une lettre dans le string.
. ajout d'une lettre dans le string.

exemple : seingerrie singerie en 2 suppressions
seingerrie seigneurie en 2 suppressions et 2 ajouts.

Les auteurs nous proposent ensuite des algorithmes pour opérer les transformations nécessaires.

Il reste à attribuer un poids aux diverses opérations élémentaires selon l'importance qu'on leur accorde.

L'intérêt de cette démarche est que l'on pourrait calculer une distance entre le mot erroné et chacun des mots connus. On proposerait alors comme correction le mot dont la distance à un mot connu est la plus courte.

2. A statistical method of spelling correction.

Si la première méthode rejoignait nos idées sur la question, celle-ci nous a bien surpris.

Quand un mot n'est pas écrit comme l'ordinateur le veut, il faut le modifier et voir s'il accepte cette nouvelle forme.

Mais, par où commencer ? Par la première lettre ?
Par la dernière ?
L'idée de Ronald W. Cornew est de modifier d'abord les lettres qui offrent la plus grande probabilité d'être fautive.

L'auteur a observé les mots et il a remarqué que certaines séquences de lettres étaient bien plus fréquentes que d'autres. Il va alors recenser dans son langage, l'anglais, un échantillon représentatif et se constituer une matrice. Il l'appelle matrice de digrammes. Un digramme est une suite de deux lettres.

Voici cette matrice, regardez-là, vous découvrirez aisément comment l'interpréter.
Sachez seulement que chaque valeur représente un nombre d'observations parmi 10.000 digrammes.

LETTER OF THE SECOND POSITION

LETTER OF THE FIRST POSITION	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	SPACE
A	1	32	39	15		10	18		16		10	77	18	172	2	81	1	101	67	124	12	24	7		27	1	49
B	8				58				6	2		21	1		11			6	5		25				19		2
C	44		12		55	1		46	15		8	16			59	1		7	1	38	16		1				7
D	45	18	4	10	39	12	2	3	57	1		7	9	5	37	7	1	10	32	39	8	4	9		6		222
E	131	11	64	107	39	23	20	15	40	1	2	46	43	120	45	32	14	154	145	80	7	16	41	17	17		446
F	21	2	9	1	25	14	1	6	21	1		10	3	2	38	3		4	8	42	11	1	4		1		99
G	11	2	1	1	32	3	1	16	10			4	1	3	23	1		21	7	13	8		2		1		50
H	84	1	2	1	251	2		5	72			3	1	2	46	1		8	3	22	2		7		1		68
I	18	7	55	16	37	27	10				8	39	32	169	63	3		21	106	88		14	1	1		4	2
J					2										4						4						
K					28				8					3	3				2	1			3		3		17
L	34	7	8	28	72	5	1		57	1	3	55	4	1	28	2	2	2	12	19	8	2	5		47		49
M	56	9	1	2	48			1	26				5	3	28	16			6	6	13		2		3		40
N	54	7	31	118	64	8	75	9	37	3	3	10	7	9	65	7		5	51	110	12	4	15	1	14		135
O	9	18	18	16	3	94	3	3	13		5	17	44	145	23	29		113	37	53	96	13	36		4	2	83
P	21	1			40			7	8			29			28	26		42	3	14	7		1		2		13
Q																					20						
R	57	4	14	16	148	6	6	3	77	1	11	12	15	12	54	8		18	39	63	6	6	10		17		95
S	75	13	21	6	84	13	6	30	42		2	6	14	19	71	24	2	6	41	121	30	2	27		4		274
T	56	14	6	9	94	5	1	315	128			12	14	8	111	8		30	32	53	22	4	16		21		196
U	18	5	17	11	11	1	12	2	5			28	9	33	2	17		49	42	45				1	1	1	5
V	15				53				19						6												
W	32		3	4	30	1		48	37			4	1	10	17	2		1	3	6	1	1	2				7
X	3		5		1				4						1	4				1	1						3
Y	11	11	10	4	12	3	5	5	18			6	4	3	28	7		5	17	21	1	3	14				132
Z					5				2			1														1	
SPACE	247	85	94	91	55	84	25	95	118	14	4	45	79	43	142	82	1	62	138	363	28	8	127	1	15	1	

matrice de digrammes

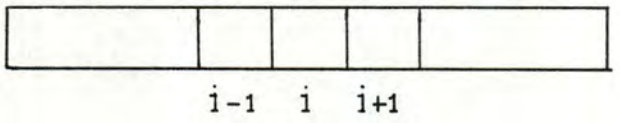
Nous constaterons ensemble qu'en anglais le digramme "TH" apparaît en moyenne 315 fois sur 10.000, le digramme JA n'apparaît quasi jamais, le digramme "SQ" revient seulement 2 fois sur 10.000, 556 mots anglais sur 10.000 se termine par "E"(c'est la fréquence du digramme "E ").

Deux questions se posent : comment va-t-il choisir la lettre qu'il va modifier en premier ? Par quelle autre lettre va-t-il éventuellement la remplacer ?

première question :

La lettre qui sera la plus probablement fausse sera celle qui offrira à la fois la probabilité la plus faible de suivre celle qui la précède et de précéder celle qui la suit !

soit un string de n caractères alphabétiques.



M - une matrice digramme, ses indices sont les caractères alphabétique + le blanc dans l'ordre ASCII.

Pour tout i variant de 1 à n, il correspond deux valeurs distinctes :

$V1i = M\left[S_{[i-1]}, S_{[i]}\right]$

qui représente la fréquence du digramme formé du caractère S[i] et de son voisin de gauche.

$V2i = M\left[S_{[i]}, S_{[i+1]}\right]$

qui représente la fréquence du digramme formé du caractère S[i] et de son voisin de drcite.

Le produit Pi = V1i X V2i sera d'autant plus petit que les suites de caractères (S[i-1],S[i]) et (S[i],S[i+1]) sont peu observées.

La lettre qui est la plus vraisemblablement fausse est celle qui présente le plus petit produit Pi.

L'auteur choisit donc S[i] tel que $Pi = \min_{j=1..n} Pj$

seconde question : Par quelle lettre remplacer celle qu'on enlève ?

Ronald W. Cornew emploie la même méthode. Il va choisir la lettre qui offre la plus grande probabilité de se retrouver entre les 2 lettres S[i-1] et S[i+1].

Ceci se traduit par une consultation de la matrice M.

La ligne $M[S[i-1],*]$ nous donne la liste des lettres qui peuvent suivre $S_{[i-1]}$ et leur fréquence.

la colonne $M[*, S[i+1]]$ nous donne la liste des lettres qui peuvent précéder S[i+1] et leur fréquence.

Le produit $P^* = M[S[i-1],*] \times M[*, S[i+1]]$ représente la fréquence.

de la séquence de caractère S[i-1],*, S[i+1]

L'auteur choisira donc le caractère * tel que $P^* = \max_{j=1..n} P_j$,

c'est pour lui que la probabilité conjointe de suivre S[i-1] et deprécéder S[i+1] est la plus grande.

Pour terminer son article, l'auteur nous donne des statistiques pour montrer l'efficacité de sa méthode. Retenons pour mémoire que dans un test de correction orthographique portant sur un vocabulaire de 1000 mots il a bien corrigé l'orthographe.

739 fois il a bien corrigé l'orthographe
241 fois, le programme a pris un mauvais chemin, il aurait dû garder les lettres les moins fréquentes.
20 fois, il est tombé sur un autre mot valide.

C. Solution retenue.

1. Introduction.
2. la solution.

1. introduction.

Rappel de l'objectif.

L'objectif sera de trouver dans un minimum de temps le mot ressemblant le plus à ce que l'enfant aurait voulu écrire. Un temps de réponse de 5 secondes nous semble une bonne moyenne. Dix secondes est une valeur à ne dépasser qu'exceptionnellement.

Présentation des contraintes : elles sont deux.

D'abord, il y aura énormément de calculs. Le vocabulaire, même réduit, comporte encore 280 mots et il faudra fatalement beaucoup de comparaisons pour en extraire une bonne proposition de correction.

Ensuite, il faut savoir que nous travaillons sur une petite machine. Les écoles primaires n'ont pas le moyen de s'offrir des outils de plusieurs millions. Ce matériel sur lequel s'effectuera le projet influencera le choix d'une solution de manière déterminante.

Il s'agit d'un micro-ordinateur de 16 bits APPLE II, de 48 K de mémoire centrale. Le support secondaire est un disque souple 5 1/4", il est découpé en 280 blocs de 0,5 K et sa capacité totale disponible est de 137 K.

2. la solution.

La solution que nous avons retenue emprunte les idées exposées dans les articles précédents.

a. la solution rejetée.

La première était de calculer une distance entre le mot donné et chacun des mots de la liste. On choisissait alors dans la liste le mot qui présentait le plus petit écart au nom donné. Le critère de distance était celui présenté dans le premier article.

Les avantages de cette solution :
- elle trouve toujours une proposition de correction et mesure l'écart.
- l'algorithme correspondant serait très court
- le critère de distance qui se base sur une étude de position des lettres est très éloigné d'un raisonnement humain (qui se pencherait d'abord sur des ressemblances phonétiques ou sur une application intuitive de règles propres au langage)

Il se détache donc de critères plus familiers et nous présente le problème sous un angle favorable à la formalisation informatiq

On peut quand même remarquer au passage que ce critère "position des lettres" englobe le critère phonétique : dans notre langue française, la majorité des sons sont groupés sur deux ou trois lettres. Ecrire un mot comme on l'entend n'engendrera donc le plus souvent que des différences de une ou deux lettres.

Ce critère offre aussi, de par sa nature, la possibilité de récupérer les fautes de frappe, fautes qui n'obéissent à aucun raisonnement humain.

Malheureusement, le temps de réponse est directement proportionnel à la longueur de la liste du vocabulaire debase. Céci engendre pour 280 mots des temps de réponse de l'ordre de 40 secondes. C'est inacceptable. Ells sera donc réservée soit à des applications sur du matériel plus puissant, soit à des applications sur des listes très courtes.

b. solution retenue.

Il fallait donc employer une méthode qui n'impose pas un parcours complet de la liste de vocabulaire.

L'idée est de partir du mot fautif et de le transformer, caractère par caractère, pour aboutir à un mot existant dans la liste. Cependant, nous ne pouvons pas raisonnablement imaginer de composer l'ensemble des combinaisons possibles, la longueur moyenne d'un même mot est de 7 lettres et l'alphabet en compte 26. Ce n'est pas pensable!

C'est ici que la méthode statistique exposée dans le 2è article va nous aider. Elle va nous permettre d'orienter nos recherches vers les solutions les plus probales.
Cette méthode possède quand même trois désavantages :

. nous devrions faire l'hypothèse qu'il n'y a qu'une seule faute dans le mot.
Tenir compte du fait qu'il y a peut être deux ou trois fautes nous entrainerait dans des combinaisons beaucoup trop nombreuses

. conséquence directe, on ne reconnaîtrait pas un mot s'il y avait plus d'une faute.

. les mots longs sont défavorisés.
D'une part, ils offrent une plus grande probabilité de double faute, ensuite, s'il y a une ou deux fautes à la fin d'un mot long, il faudra beaucoup de calculs et donc beaucoup de temps pour corriger ou dire qu'on ignore le mot.

Nous allons malgré tout l'employer comme base de notre raisonnement. Pour accroître les programmes en temps de réponse, et la capacité de recouvrement d'erreur, nous allons la modifier et la marier avec un algorithme de distance.

1. les données.

les données nécessaires :

. le vocabulaire.

Il sera trié alphabétiquement et d'accès rapide. De plus les longs mots (plus de 6 lettres) seront triés alphabétiquement sur leurs trois dernières lettres et accessibles rapidement.

Les mots seront assez distincts les uns des autres.
ex. : si j'ai 'mouche', je rejette 'mouchard' et 'mouchoir'.
Cela n'est pas nécessaire théoriquement, mais cela nous simplifiera la tâche. cf p55 algorithme de distance.

. la matrice de fréquence des digrammes - cf p47

Elle ne donnera pas les fréquences de digramme d'une langue, mais les fréquences des digrammes du vocabulaire précis sur lequel s'exécutera le système. Elle sera également d'accès rapide.

. un mot fautif à corriger.

2. Traitement.

Dans notre solution, nous distinguerons les mots courts et les mots longs.

. les mots courts.

hypothèses: au plus 6 lettres
une seule faute par mot.

traitement :

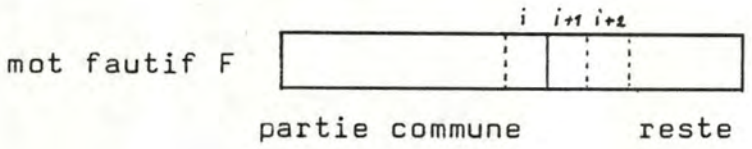
Toute la logique du traitement repose sur la division du mot en deux parties : une partie commune et un reste.
Nous vous expliquons immédiatement.

soit un mot erroné	OISAU	Il ne figure pas sur la liste.
	OIE	
insérons le	OISAU	
	OISEAU	

Il possède toujours un voisin, presque toujours deux.
Prenons les premières lettres de ces mots et considérons la plus grande partie qu'il puisse avoir en commun avec l'un d'eux. Dans l'exemple, c'est "OIS".

Nommons la "partie commune". Le reste du mot s'appelant

"reste" tout simplement.



Selon notre hypothèse, il y a une seule faute dans le mot. Cela fait deux cas à envisager.

La partie commune est bien le début du mot que voulait écrire l'enfant.

ex. il voulait écrire "OISEAU" et a écrit "OISAU".

Alors, une erreur se trouve forcément à la jonction de la partie commune et du reste; entre F [i] et F [i+1]

Essayons d'intervertir F [i+1] et F [i+2]

Essayons de supprimer F [i+1]

Essayons de remplacer F [i+1] et employons les statistiques de la matrice de fréquence des digrammes.

Essayons d'ajouter entre F [i] et F [i+1] et employons les statistiques de la matrice de fréquence des digrammes.

Il se peut que ces opérations ne nous apportent rien.

C'est peut-être parce qu'il y a plus d'une erreur dans le reste du mot.

ex : l'enfant aurait écrit OISAUU.
Par hypothèse, nous abandonnons cette voie.

Envisageons le second cas.

La partie commune du mot de notre écolier est commune avec le début d'un autre mot que celui qu'il voulait vraiment écrire.

ex. : notre petit sujet aurait écrit OASEAU

dans la liste cela donnerait OASEAU
OASIS
OISEAU

On pourrait croire qu'il a voulu écrire OASIS, mais comme cela ne marche pas, on va vérifier s'il n'ya pas une erreur dans la partie commune.

Essayons de modifier la partie commune. Essayons d'abord les opérations bon marché en temps de calcul.

l'inversion de chacune des lettres de la partie commune de proche en proche avec la suivante.

AOSEAU
OSAEAU
OAESAU cela ne donne rien.

la suppression :

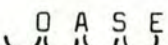
ASEAU
OSEAU
OAEAU cela ne donne rien non plus... aie.aie.aie !

Essayons alors de remplacer une lettre. Aidons-nous de la méthode statistique.

2 questions :

. quelle est la lettre qui semble le moins à sa place ?

consultons les fréquences de digrammes (nous donnons des CHIFFRES au hasard, car nous n'avons pas fait d'états imprimés).



5 1 3 10 fréquences

5 3 30 probabilités conjointes.

C'est le A qui est statistiquement le plus mal logé entre ses voisins.

. Par quelle lettre remplacerions-nous le A ?

Consultons la matrice de digramme et effectuons le produit

ligne O x colonne S.

Nous trouvons un vecteur de 27 éléments. Comme le hasard fait bien les choses, c'est le I qui présente la plus grosse probabilité.

Remplaçons le A par le I et attrapons notre OISEAU.

Si le remplacement n'avait pas satisfait, nous aurions envisagé d' ajouter une lettre dans la partie commune.

Un nouvel échec nous aurait amené à la conclusion qu'il y a plus d'une faute dans le mot. Nous aurions dû l'abandonner.

. les mots longs.

Hypothèse.

. il possède plus de 6 lettres.

. il peut contenir plusieurs fautes à condition que les 4 premières lettres ou les 4 dernières lettres soient justes.

Traitement.

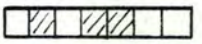
Le traitement repose sur l'idée que si l'enfant a écrit un mot dont le début ou la fin ressemble suffisamment à celui ou celle d'un mot existant, c'est que c'est bien ce mot que notre petit ami a voulu écrire.
Ici, quatre lettres communes nous semblent suffisantes au début d'un mot, le vocabulaire étant composés de mots très distincts. A la fin du mot, trois lettres communes suffiron
Les fins de mots étant plus variées que les débuts (lisez P57 la 3eme remarque)
Si malgré tout nous rencontrons plusieurs mots candidats, ils seront peu nombreux et l'algorithme de distance tombera à point pour les départager.

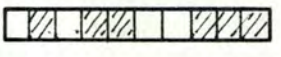
Présentation de l'algorithme de distance.

données.

deux strings dont les longueurs peuvent être différentes. l'un est le string d'entrée. C'est un mot mal orthographié. l'autre est un string de base. Il appartient à la liste de vocabulaire.

Algorithme :

soit Entrée E_i 

Base B_j 

les zones hachurées représentent les caractères de E et de B qui diffèrent.

L'algorithme présenté tiendra la comptabilité de ces différences. Comme nous vous l'avons déja dit, ces différences sont dues à des substitutions, des oublis et des ajouts de lettres. Ici, nous ajouterons les inversions.

Le nombre de différences constitue la distance entre les deux strings.

Supposons que les i-1 et j-1, premiers caractères des deux strings aient déjà été comparés.

. une inversion coûte un point.

Attention ! une inversion n'est pas équivalente à deux substitutions.

Nous aurons une inversion si

$$(E[i] \neq B[j]) \wedge (E[i] = B[j+1]) \wedge (E[i+1] = B[j])$$

. une substitution coûte un point.

Il y a substitution si

$$(E[i] \neq B[j]) \wedge (E[i+1] = B[j+1])$$

. un oubli coûte un point.

Il y a oubli si

$$(E[i] \neq B[j]) \wedge (E[i] = B[j+1]) \wedge (E[I+1] \neq B[j])$$

. un ajout coûte un point.

il y a ajout si

$$(E[i] \neq B[j]) \wedge (E[i] \neq B[j+1]) \wedge (E[i+1] = B[j])$$

quelques exemples :

soit le string de base MARSUPILAMI.

1ère entrée : MARSPUILAMI.
une inversion, la distance égale 1

2ème entrée : MARSUPINAMI
une substitution, la distance égale 1

3ème entrée : _ARSUIPLAMI

un oubli + une inversion, la distance égale 2

4ème entrée : MARRUPLAM_

une substitution+ 2 oublis, la distance égale 3.

Il reste à établir deux listes de mots.
Dans l'une, les quatre premières lettres des mots sont communes à celles du mot litigieux.
Dans l'autre, ce sont les trois dernières lettres qui sont communes.
Parmi ces deux listes, nous allons rechercher le mot dont l'écart au mot litigieux est le plus petit, ceci grâce à l'algorithme de distance.

Après les idées de solution, la bibliographie, l'exposé de la solution retenue, discernons les problèmes futurs

d. problèmes ultérieurs

Jusqu'ici, nous avons supposé que les mots étaient isolés, invariables.
Des problèmes surviendront quand les mots à corriger appartiendront à des phrases.
Rappelons-nous que la majorité des mots ont une terminaison variable . Ils possèdent plusieurs orthographes (jusqu'à 50 pour un verbe !).

Trois types d'obstacles entravent notre démarche.

. Pédagogiquement, il sera intéressant de distinguer les fautes d'usage et les fautes d'accord. Ce problème sera surmonté par un raisonnement adéquat. Par exemple, en considérant qu'on connait un mot quand on a identifié un radical invariable.

. Le nombre croissant de comparaisons va augmenter le nombre d'accès à la base de données et diminuer sensiblement les performances. Ce deuxième obstacle nous effraye assez. Il faudra envisager une solution plus technique. Peut-être réécrire le programme, en tout ou en partie, en langage Assembleur.

. Le troisième ennui qui se présentera sera une modification de l'idée de solution.
Souvenez-vous : pour les mots longs nous supposons que si la fin du mot de l'élève correspond à la fin d'un mot de notre dictionnaire, c'est ce dernier que l'enfant à voulu écrire.

Si nous tenons compte des accords, cette hypothèse devient audacieuse, beaucoup de mots ont la même finale (ex. les verbes à la 1ère personne du pluriel se terminent tous par ions à l'imparfait).

rappel a. la correction automatique des fautes d'orthographe d'usage et des fautes de frappe.

b. la génération des phrases.

b.génération des phrases.

Comme je vous l'ai déjà dit, le générateur est un programme assez simple à écrire si l'on dispose :

. d'une bonne base de données, bonne quant à son contenu (choix des mots, découpe en classes sémantiques) et à son organisation (économie de mémoire et rapidité d'accès).

. de diagrammes syntaxiques pour déterminer les phrases à générer. (CF ANNEXE A.I)

Je vous ai décrit de mon mieux comment j'ai défini l'un et l'autre.

Le rôle du générateur dans le projet global est essentiellement pédagogique. Il donne des exemples. Pendant la réalisation, il nous permettra de juger la valeur de la base de données.

J'ai quand même anticipé l'emploi qui sera fait du générateur. Ainsi, je me suis dit que l'on pourrait un jour avoir envie de jouer avec ces phrases (les pronominaliser, changer la conjugaison, les mettre sous forme interrogatives ...).

Plutôt que de les imprimer directement à l'écran, j'ai préféré d'abord enregistrer ces phrases avec leur structures.

Pour le reste de ce qui concerne la génération, je préfère vous renvoyer aux annexes (pAII) où vous trouverez toute la documentation.

Retenez simplement que la technique de génération de phrases se déroule en trois étapes, plus une qui est manuelle

. choisir les paramètres de la phrase à constuire
. construire la phrase en parcourant un chemin dans les diagrammes syntaxiques
. enregistrer la phrase et sa structure, l'imprimer à l'écran
. sourire ...

TROISIEME PARTIE : L'IMPLEMENTATION

A. COMMENT SE PRESENTENT LES DONNEES A ORGANISER ?

B. ORGANISATION DES DONNEES RELATIVES A LA SYNTAXE

C. ORGANISATION DES DONNEES RELATIVES A LA SEMANTIQUE

D. MATRICE DES DIGRAMMES

III. L'IMPLEMENTATION

Je ne veux pas vous ennuyer et vous donner ici les choix d'implémentation de chacun de mes programmes. Ils seront entierement spécifiés et documentés en annexe.

Je voudrais seulement présenter l'implémentation de la base de données.

L'emploi d'une machine relativement lente et petite pour l'usage que j'en fais m'a contraint à optimiser au maximum l'occupation de mémoire par les données. Ceci au détriment de la lisibilité.

Petit plan.

A. comment se présentent les données à organiser.
B. organisation des données relatives à la syntaxe.
   . sur fichier
   . en mémoire principale.
C. organisation des données relatives à la sémantique.
D. matrice des digrammes.

A. Comment se présentent les données à organiser ?

Selon un premier axe, je distinguerai :

les données permanentes.

. le vocabulaire retenu (noms communs, adjectifs qualificatifs, verbes,noms propres,adverbes).
. les différents accords de ces mots.
. la découpe en classes sémantiques.

les données calculées.

les valeurs de la matrice de digrammes. Elle dépend du vocabulaire choisi Elle change si le vocabulaire change.

Selon un autre axe, je séparerai les données qui nécessitent :

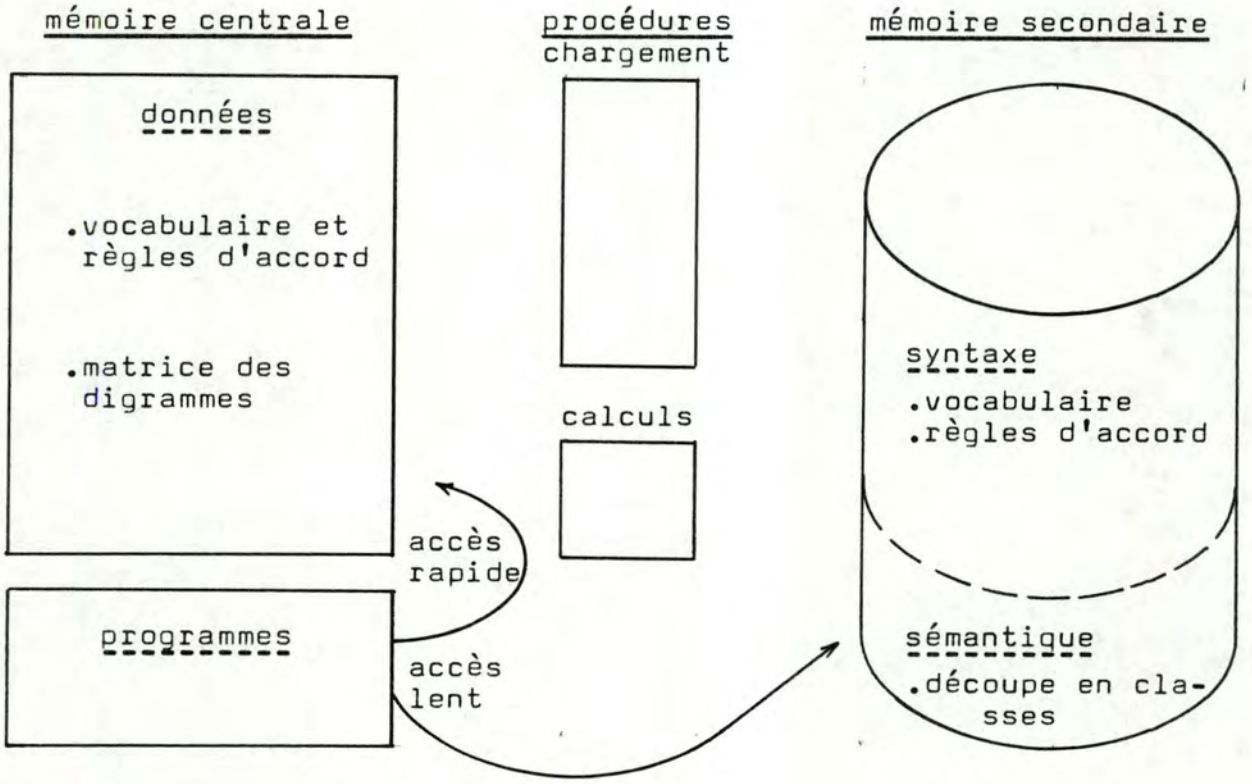
. un accès extrèmement rapide.

  Le nombre d'accès à ces données entre deux sorties sur écran est très élevé et ne doit pas trop diminuer le temps de réponse.

  Ce sont les données concernant le vocabulaire, les règles d'accord la matrice de digrammes. Elles seront en mémoire centrale pour rendre l'accès physique beaucoup plus rapide.

. celles qui n'exigent pas un accès si rapide.

Il n'y a que 4 ou 5 accès à ces informations entre 2 sorties à l'écran et le temps de réponse est peu menacé. Elles resteront donc uniquement sur mémoire de masse.



Il faudra donc :

a. décider d'une représentation des données permanentes sur support secondaire.

b. décider d'une représentation en mémoire centrale des données nécessitant un accès rapide et écrire une procédure qui charge ces données à partir des fichiers.

c. choisir une représentation de la matrice de digrammes et écrire une procédure qui calcule cette matrice à partir du vocabulaire existant.

Voici d'abord la syntaxe. Je souhaite vous présenter l'organisation des données sur disque et ensuite en mémoire centrale. Viendra ensuite l'organisation sémantique qui n'existe que sur disque.

Pour clôturer ce 3è chapitre, je toucherai deux mots de la matrice de digrammes.

B. Organisation des données relatives à la syntaxe.

Ces données sont permanentes. Elles seront chargées en mémoire centrale durant l'exécution du programme pour accroitre les performances. Il s'agira de ne pas gaspiller les 48 k disponibles.

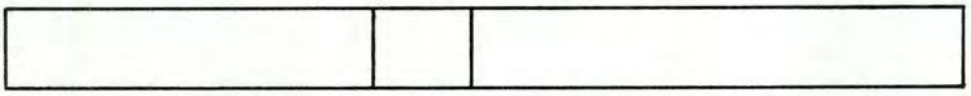
Voyons d'abord l'organisation sur fichiers. Nous étudierons ensuite l'aspect de ces données en mémoire principale.

a. sur fichier.

Comme déjà dit, un mot est formé par la concaténation de son radical invariable et de sa terminaison variable.

Une idée serait d'associer dans un enregistrement un radical et la liste des terminaisons permises selon la nature du mot.

Un enregistrement aurait cette forme



partie invariable du mot	nature du mot	liste des terminaisons possibles du mot selon sa nature

comme la taille de la liste des terminaisons varie beaucoup selon la nature du mot concerné, l'emploi du record PASCAL à types variables (case of) quoique très lisible, n'est pas intéressant.
Il faut savoir qu'il est implémenté en longueur fixe.
PASCAL ucsd réserve pour chaque enregistrement la plus grande taille dont il puisse avoir besoin . Dans notre cas, cela fait un beau gaspillage, les adverbes et les noms propres n'ayant pas de terminaison.

La solution que j'ai retenue est de dissocier la liste des terminaisons de l'enregistrement et de l'éclater en autant de types d'enregistrements qu'il y a de nature de mots ayant des terminaisons.

Cela nous donne :

FICHPREFIXE : file of WORD ;

Ce fichier contient tous les radicaux du vocabulaire et leur nature (*).

Pour faciliter la recherche, il sera trié par ordre alphabétique chaque enregistrement de ce fichier aura la forme

WORD

partie invariable du mot — nature du mot

```
WORD = record
          ORTHO:string 20 ;
          NAT  :NATURE
       end;
```

FSTXNCOM : file of STXNCDESC ;

Ce fichier contient pour chaque nom commun un indicateur qui lui donne quelques caractéristiques - admet-il une forme - féminin singulier, pluriel, masculin singulier, pluriel ?

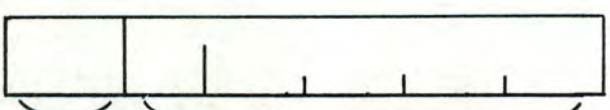
admet-il un adjectif qualificatif avant, après lui ?

Cela peut vous surprendre que cette information sémantique se mèle à des informations syntaxiques. Vous avez raison. Mais chaque fois que je génère une phrase, ces deux informations sont consultées ensemble. Je gagne donc un accès disque en les groupant.

Il contient aussi les différentes orthographes des terminaisons féminin singulier et pluriel, masculin singulier et pluriel si elles existent.

Ce fichier est trié selon l'ordre alphabétique des radicaux des mots qui possèdent ces terminaisons.

Chaque enregistrement de ce fichier aura la forme



caractéristique du nom commun — suffixe

(*) Par abus de langage, j'identifie un radical au mot auquel il appartient.

```
SUFFIXE = packed record.
                        FSY : Boolean;
                        FPY : boolean;
                        MSY : boolean;
                        MPY : boolean;
                        FS  : String (8);
                        FP  : string (8);
                        MS  : string (8);
                        Mp  : string (8)
                end;

STXNCDESC = packed record

                        CARACT : array [1..6] of boolean;
                        TERM   : SUFFIXE

                end;
```

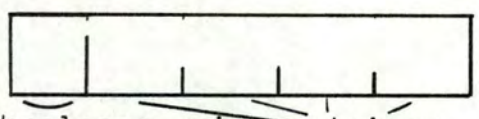
Il y a redondance dans ces informations. Je répète qu'un nom commun a ou n'a pas de forme féminin singulier, féminin pluriel, masculin singulier, masculin pluriel.

Cette erreur provient uniquement de ma maladresse et de la complexité des problèmes à résoudre. Elle n'a aucune conséquence et je n'ai pas cherché à la corriger.

```
FSTXADJ : file of SUFFIXE ;
```

Ce fichier contient pour chaque adjectif qualificatif les différentes orthographes de terminaisons féminin singulier et pluriel, masculin singulier et pluriel. Il est trié selon l'ordre alphabétique des radicaux et des adjectifs qui possèdent ces terminaisons.
Le suffixe est déjà décrit pour les noms communs. C'est le même. Chaque enregistrement du fichier aura donc la forme

SUFFIXE



F*STXVb : file of SUFFVB ;

Puisque les verbes sont divisés en 3 groupes absolument imperméable (E-IS-S), je vais ma simplifier la tâche en créant un type de fichier pour chacun.

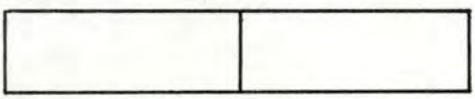
Leurs structures sont absolument identiques. Ce sont : FESTXVB, FISSTXVB, FSSTXVB.

Dans chaque enregistrement de ce fichier nous trouverons :

. la terminaison infinitive
. un numéro de conjugaison.

Ces fichiers sont triés selon l'ordre alphabétique des radicaux des verbes qui possèdent ces terminaisons.

SUFFVB



infinitive numéro de conjugaison

```
SUFFVB = record

           INF :string[5];
           CONJ:integer

         end;
```

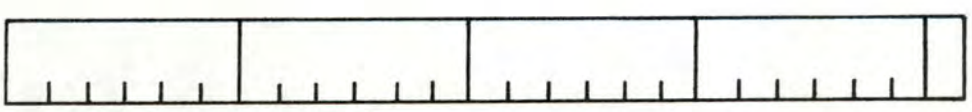
Vous savez que j'ai réparti les verbes en 3 groupes E-IS-S, mais au sein de ces groupes, il reste des différences dans la façon de conjuguer.
J'ai donc pris le guide Bescherelle - l'art de conjuguer - de 8000 verbes. Pour chacun de mes verbes, j'ai répertorié les formes de conjugaison. J'ai répéré 18 paradigmes pour 44 verbes. Je les ai numéroté de 1 à 18 (correspondance avec les numéros Bescherelle en annexe pAXI).

Pour chaque enregistrement SUFFVB d'un des trois fichiers, on trouve ce numéro dans le champ CONJ.

Il ne reste qu'une question. Comment sont enregistrées les différentes conjugaisons pour les temps de l'indicatif et les personnes considérés?

J'ai créé un septième fichier : TERMCONJ : file of GROUPE ;

Chaque GROUPE est un enregistrement en 5 parties.



présent imparfait passé simple futur simple participe passé

Chaque partie est elle-même divisée en 6 zones qui représentent respectivement :

1è,2è, 3è personne singulier, 1è,2è,3è personne pluriel.

```
TABSTRING = packed array [1..6] of string[8];
GROUPE = packed record

                PRESENT     :TABSTRING;
                IMPARFAIT   :TABSTRING;
                PASSESIMPLE:TABSTRING;
                FUTURSIMPLE:TABSTRING;
                PARTPASSE   :string[4]

              end;
```

Je vous ai donc montré l'implémentation des informations syntaxiques sur fichiers.

Toutes ces informations doivent aussi être chargées en mémoire principale.

b. en mémoire principale.

Reprenons dans le même ordre les différents éléments à charger.

FICHPREFIXES. Son contenu sera donc chargé en mémoire principale.

Voici deux solutions:

. garder une implémentation parallèle à celle que nous connaissons déjà sur fichier, c'est à dire remplacer les fichiers par des tableaux. C'est la méthode la plus simple.
Elle n'est pas la plus économique.
Comparons son prix au coût minimum théorique.

La longueur d'un mot, en moyenne, est 7 lettres. Il y a 250 mots.
1750 caractères ou 1,75 Kbyte devraient nous suffire.
L'emploi des tableaux nous forcera en longueur fixe à retenir la longueur du plus long mot : 18 lettres. Cela couterait 18 x 250 caractères, soit 4,5 kbyte, soit 2,8 kbyte de perdus.
Quel gâchis !

Monsieur Cherton nous a soufflé, à Jacques Kinet età moi, le truc pour la récupérer.

" Vous pourriez employer un système d'indirection, dit-il.
" Dans un grand tableau de caractères compactés, vous enfileriez
" tous vos mots bout à bout, par ordre alphabétique mais séparés
" par un caractère bidon. Un qu'on n'emploie pas en FRANÇAIS.

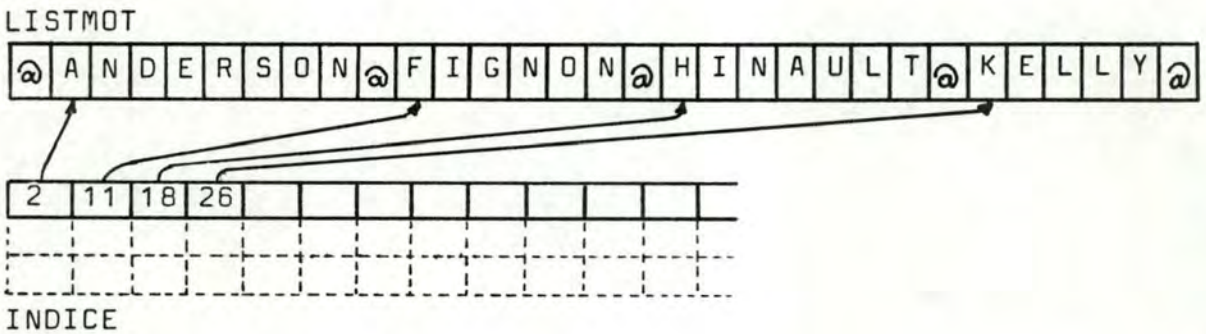
... et alors, pour référencer les mots ?

" vous pourriez vous créer un tableau de pointeurs. Il aura une
" entrée par mot. Chaque valeur vous dira à quel indice commence
" le mot dans le tableau compacté.

Ainsi nous irions chercher le mot lettre par lettre à partir de cett adresse et le caractère spécial nous indiquerait la fin du mot ?

" c'est ça!".

Nous avons adopté cette solution.



En fait, le tableau INDICE ne contient pas que l'adresse du début du mot.
On y trouvera encore 2 types d'information.

. la nature du mot.
nom propre féminin, nom propre masculin, nom commun, adjectif qualificatif, adverbe de manière, verbe en E, en IS, en S.

Vous remarquerez que les types d'articles NOM PROPRE et RADICAL-VERBE (cf modèle d'accès p.41) ont été éclatés en 2 et 3 types d'articles. Dans le programme de génération, lorsque j'ai besoin d'un nom propre ou d'un verbe, je dois préciser le genre ou le groupe E - IS - S du mot désiré. Je favoriserai donc un accès direct en les groupant selon ces propriétés.

- adresse dans une table.

Puisque dans ce tableau Listmot, il n'y a que des radicaux de mots, il faut bien dire où sont les terminaisons correspondantes.

A chaque nature de mot, j'associe une table. Elle décrit ces terminaisons s'il y en a.

Il y a donc pour chaque mot une référence vers une de ces tables.

Etudions ces différentes tables.

Elles concernent des mots variables ou invariables.

Pour les mots variables :

TNCOM (table des noms communs), TADJQ (table des adjectifs qualificatifs)
TEVERBE, TISVERBE, TSVERBE (tables des verbes en E, en IS ou en S)

Chacune de ces tables contient 2 types d'information.

. une copie conforme du fichier correspondant (FSTXNCOM, FSTXADJ, FESTXVB, FISSTXVB, FSSTXVB)

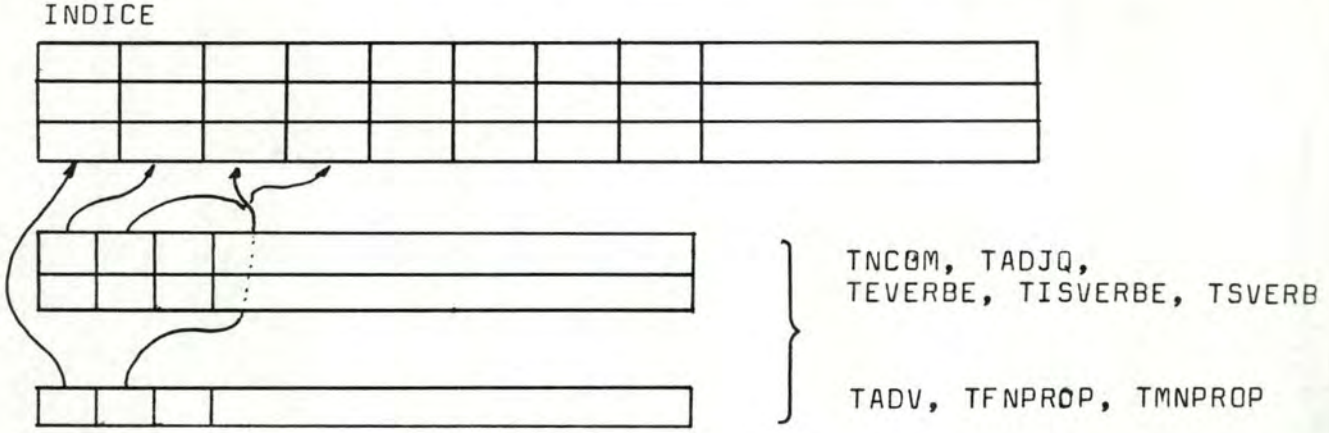
. le numéro d'ordre alphabétique du mot, c'est à dire un pointeur vers le tableau INDICE, vers son radical.

l'ordre des éléments de chaque table est celui du fichier correspondant.

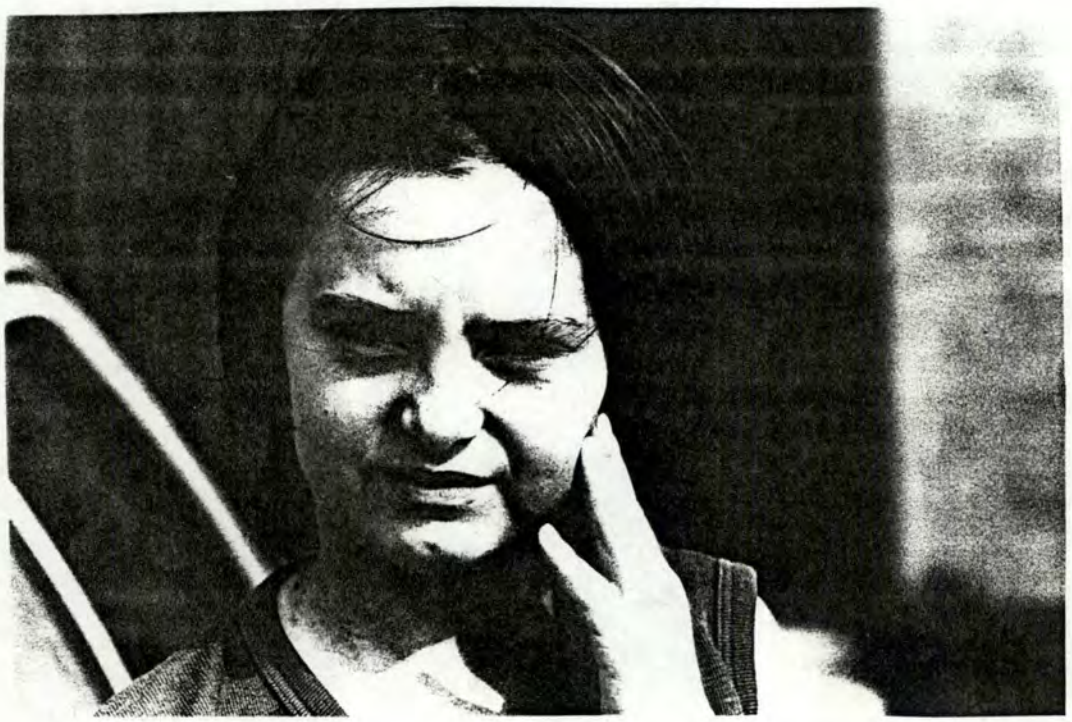
Pour les mots invariables.

TADV (table des adverbes de manière,) TFNPROP, TMNPROP (tables des noms propres féminin masculin.

La seule information quel'on y trouvera pour chaque mot est son numéro d'ordre abphabétique, c'est à dire une référence vers le tableau indice.



Pour chaque table, quelle que soit la nature du mot qu'elle concerne,son indice est la clé d'accès que j'ai indiquée dans le schéma d'accès p . En fait,c'est le numéro d'ordre alphabétique d'un mot parmi les autres mots de même nature.



Revenons à TEVERBE, TISVERBE, TSVERBE.

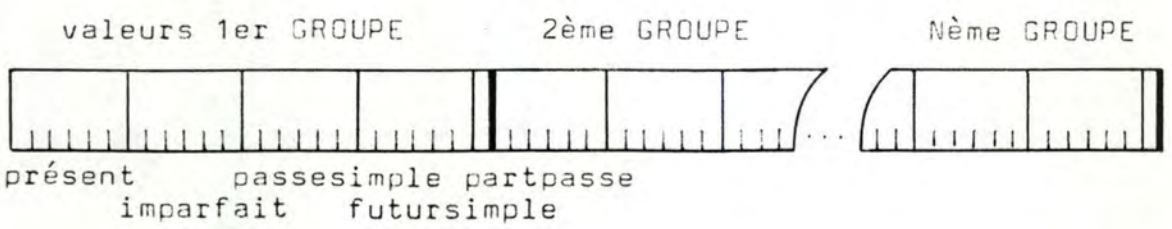
Nous avons vu quepour les verbes, nous n'enregistrions pas les terminaisons mais plutôt un numéro de référence vers des règles de conjugaison.

Il me reste à vous décrire comment sont enregistrées ces règles en mémoire principale.

Je répète que l'unité d'enregistrement sur fichier est le GROUPE (cf p.64)

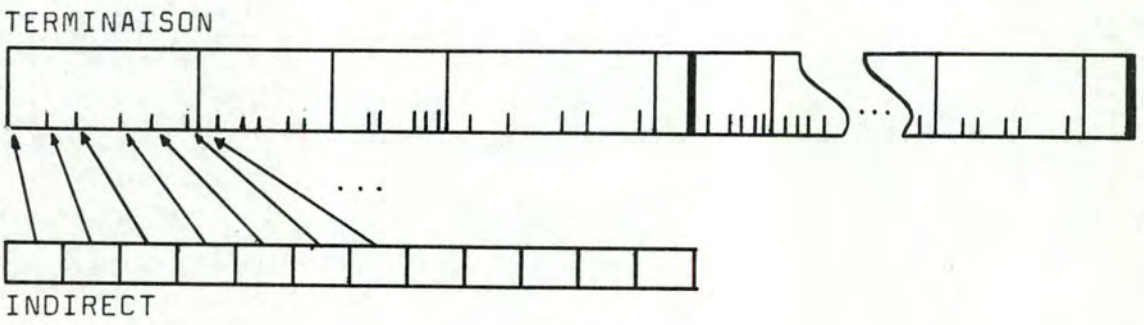
Je vais charger bout à bout tous les composants de chaque groupe du fichier dans un tableau compacté de caractères (cela ne vous rappelle rien, cette astuce pour économiser de la place ?)

Voici ce tableau - TERMINAISON.



Mais les terminaisons sont de différentes longueurs.

Je vais donc créer un tableau de pointeurs pour connaître le début de chaque zone logique de ce tableau



exemple d'emploi.

Quelle est la terminaison 3è pers.singulier au futur simple d'un verbe de groupe 4 ?

Cette terminaison dans TERMINAISON, commence à l'adresse renseignée dans l'entrée.

INDIRECT [1 + 25 X (4 - 1) + 6 X (4 - 1) + 3]

car le premier indice de INDIRECT est 1 et pas 0

car dans un GROUPE il y a 25 zones logiques

car groupe 4

car on veut le début des zones du groupe 4

car il y a 6 zones logiques par temps de l'indicatif

car Futur = 4ème temps

car 3è pers du singulier

car on veut la première terminaison du FUTUR

Il faut aller chercher tous les caractères dans TERMINAISON à partir de cette adresse jusque celui qui se trouve en TERMINAISON [INDIRECT [adresse calculée+1]] exclu (c'est le début de la terminaison suivante).

C. organisation des données relatives à la sémantique.

Les accès sont relativement peu fréquents. Ces renseignements resteront sur fichiers. Ils seront au nombre de six.

FSEMADJ (fichier sémantique des adjectifs)
FSEMNCOM (f chier sémantique des noms communs)
FCLASSES (fichier sémantique des classes noms)
FSEMEVB, FSEMISVB, FSEMSVB (fichiers sémantiques des verbes en E, IS et S).

Commençons par les plus simples.

FSEMADJ.

Le schéma d'accès nous ap rend quepour chaque adjectif on doit accéder à son ou ses numéros de classe.

Le ième enregistrement de ce fichier contient l'information du ième adjectif par ordre alphabétique.

Chaque enregistrement donnera :

. le nombre de classes auxquelles appartient cet adjectif.
. la liste des numéros de classe.

Remarquons que ce fichier ne servira qu'au décodage de phrase d'enfant et n'apparaît qu'en commentaire dans mes programmes.

```
FSEMADJ = file of  SEMADJDESC;

SEMADJDESC = packed record
                    NBCL : 0..5;
                    LCLAS: array [1..3] of integer
                 end;
```

FSEMNCOM.

D'un nom commun, il faudra accéder à 3 types d'information.

- A quelle classe sémantique appartient-il ?
Pour tout enregistrement, nous avons notamment

. le nombre de classes auxquelles appartient ce nom.
. la liste des numéros de classes.

- Les numéros de classes adjectifs qu'il admet dans le sens recevoir

La question précise qui se posera sera : admet-il la classe adjectif numéro i ?

J'emploie donc une table de booleens de 5 entrées (il ya 5 classes) LCLASSE i me répondra oui ou non.

- Quels adjectifs prend-il dans le sens générer ?

On distingue des adjectifs "avant" et "après".
Pour chacun d'eux, nous retiendrons :
. Le nombre d'adjectifs (avant et après) qu'ils prennent
. La liste des numéros d'adjectif (ils sont classés par ordre alphabétique)

```
FESMNCOM = file of SEMNCDESC;

SEMNCDESC = packed record

                    GENERER    : record
                                   NBADAV:integer;
                                   NBADAP:integer;
                                   PTADAV:array[1..8] of integer;
                                   PTADAP:array[1..8] of integer
                                 end;
                    RECEVOIR   : record
                                   LCLASSE:packed array 1..5 of boolear
                                 end;
                    APPARTIENT: record
                                   NBCL :0..3;
                                   LCLAS:array[1..3]of integer
                                 end
                  end;
```

FCLASSES.

Toujours selon le schéma d'accès, nous devons, pour chaque classe sémantique connaître l'ensemble des noms communs qui la composent.

FCLASSES reprend ces informations. Ce fichier sera trié par ordre croissant de numéro de classe

Pour chaque article, nous relevons deux renseignements :

. combien de mots dans la classe
. liste des numéros de noms communs qui composent cette classe, triés par ordre croissant

```
FCLASSES = file of VALCLAS;

VALCLAS = record

            LISTEMOT:array[1..20]of integer;
            LONGLIST:integer

          end;
```

FSEMEVB, FSEMISVB, FSEMSVB.

Pour l'aspect sémantique, les verbes sont aussi disposés dans 3 fichiers selon leur groupe E, IS, S.
C'est bien plus commode de garder partout la même méthode.

A chaque verbe de groupe E,IS, ou S, il correspond un article dans un de ces trois fichiers.

Cet article contient l'information nécessaire au décodage et à la génération de phrase.

Chacun de ces fichiers est trié selon l'ordre alphabétique des radicaux des verbes qu'il concerne. Chaque article concerne donc un verbe précis.

dans le sens "recevoir une phrase"

j'enregistre - la liste des classes de noms permis pour exercer la fonction sujet, COD, COI dans une phrase écrite par l'enfant.

quand j'emploierai Cette information, ce sera pour poser la question : le verbe "untel" admet-il comme sujet COD ou COI un mot de la classe i ?

Pour faciliter la recherche de la réponse, j'ai construit un tableau compacté de booleens. Sa longueur est le nombre de classes Pour répondre à ma question, il suffit de voir si TABLEAU[i] est vrai ou faux.

dans le sens "générer une phrase"

j'enregistre
- La liste des classes de noms permis pour exercer la fonction sujet,cod ou coi dans une phrase construite par programme.

J'emploierai cette information pour connaître l'ensemble des classes parmi lesquelles je peux effectuer un tirage aléatoire.

Pour faciliter ce travail, j'emploie un tableau d'entrées.
Sa longueur est lenombre des classes retenues pour la fonction considérée (sujet, COD, COI).
Chaque élément de ce tableau me donnera un numéro de classe valable.

- La Liste des numéros des adverbes que j'ai associés au verbe

```
FSEMEVB  = file of SEMVBDESC ;
FISSEMVB = file of SEMVBDESC ;
FSSEMVB  = file of SEMVBDESC ;

SEMVBDESC = packed record

              CODEXISTE :boolean;
              COIAEXISTE:boolean;

              RECEVB    :record

                           SUJOK :record
                                    SNBCLAS:integer;
                                    STAB:packed array 1..25 of boolea
                                  end;

                           CODOK :record
                                    CODNBCLAS:integer;
                                    CODTAB:packed array 1..25 of
                                                              boolean
                                  end;

                           COIAOK:record
                                    COIANBCLAS:integer;
                                    COIATAB:packed array 1..25 of
                                                              boolean
                                  end;

                         end;

              GENERVB   :record

                           LISTSUJ:array 1..20 of integer;
                           NBSUJ  :integer;

                           ADVTAB :array 1..20 of integer;
                           NBADV  :integer;

                           COD    :record
                                     CODLIST:array 1..20 of integer;
                                     NBCOD:integer

                                   end;

                           COIA   :record
                                     COIALIST:array 1..20 of integer
                                     NBCOIA:integer
                                   end

                         end

            end;
```

D. La matrice des digrammes.

Vous avez peut-être oublié ? C'est parce que c'est déjà loin.

Je vous rappelle vite, très vite qu'il s'agit d'une base de données statistiques. Elle permet d'orienter la correction des fautes d'orthographe d'usage et des fautes de frappe.

Elle nous dit 'essaye plutôt de "chipoter" cette lettre, elle a plus de chance qu'une autre !'(pour plus détail, vous pouvez revenir à la p.)

```
TYPE MATRICE = ARRAY ['@'..'Z','@'..'Z'] of integer ;
```

Cette implémentation a comme avantage qu'elle permettra d'accéder directement à une valeur de la matrice en employant les caractères du mot à corriger comme abscisses et ordonnées.

ex. VAR DIGRAMME : MATRICE
soit le mot "groseille"

```
DIGRAMME['O','S']
```

vous donnera immédiatement le nombre de fois qu'un "O" est suivi d'un "S" dans mon vocabulaire.

QUATRIEME PARTIE : CONCLUSIONS, CRITIQUES, PROLONGEMENTS

A. CE QUE J'AI DEJA FAIT

B. CE QU'IL RESTE A FAIRE

C. LES DIFFICULTES RENCONTREES

4. CONCLUSIONS, CRITIQUES, PROLONGEMENTS

La dernière ligne droite ... Oh non ! Arrivée d'étape seulement. Une étape du Nord avec ses pavés, ses bourrasques, et ses champs de blé. Il reste la montagne, le Galibier.

Tirons quand même les leçons de ce premier tronçon. Ensuite nous parlerons du reste du parcours.

a. ce quej'ai déjà fait et les difficultés rencontrées.

Avant tout, avec Monsieur Tonneau et Monsieur Wéry, nous avons fait l'analyse d'un projet E.A.O. Le résultat est bon.
Il encourage à poursuivre la coopération enseignants - informaticiens. C'est la porte ouverte à de bons didaciciels.

Ensuite, j'ai construit un générateur de phrases.
La syntaxe n'est pas trop redoutable. Il n'est pas très difficile de construire des phrases correctes et bien accordées, si l'on n'est pas exagérément gourmand.

Notez cependant que la langue française est pleine d'exceptions. Beaucoup trop pour moi. Cela signifie qu'il y a eu parfois des "chipotages" malpropres au niveau programmation. Cela contribue à diminuer la lisibilité des programmes et à augmenter les coûts de construction et de maintenance.

Je vous ai montré aussi que l'aspect sémantique propre aux langues naturelles reste bien l'épouvantail redouté.

D'ailleurs tu as ri en lisant mes phrases. Ne nie pas, je le sais!

C'est vrai qu'elles sont drôles, boiteuses.

Vous avez compris qu'il manque le contexte dont chaque mot a besoin pour acquérir sa pleine signification.

Ex. Un instituteur habile avait promis ce vêtement bleu.

Les questions jaillissent.
Un instituteur habile ? Pourquoi pas un mécanicien, un horloger ou un bricoleur ?
Pourquoi dire qu'il est habile ? A qui a-t-il promis ? Pour quelle raison ?
C'est curieux, un instituteur ça promet des punitions, des zéros des devoirs et des leçons. Pas des vêtements bleus. Il n'est pas tailleur puisqu'il est instituteur.

C'est qu'il n'a pas de préjugés mon programme ! Il n'est pas poête non plus.

Il ne fait pas de style. La musique de phrase ? Il ignore.

N'y aurait-il pas de contacts à établir avec les linguistes ?

Pourquoi pas un mémoire avec un étudiant d'une faculté de lettres Pour mettre à jour des règles simples de choix de structures de phrases, du vocabulaire, voire des règles élémentaires de style, par exemple, j'ai cru constater qu'un adjectif sonnait mieux avant le nom, que deux mots longs successifs faisaient souvent très laid... N'est ce pas à méditer ?

N'oublions pas le système de correction automatique de fautes d'usage et de fautes de frappe que Jacques Kinet a étudié avec moi.
Je vous en ai déjà parlé suffisamment p.44 ainsi que de ses extensions nécessaires et des problèmes qui se présenteront.

b. ce qu'il reste à faire.

Ce n'est pas tout à fait la partie immergée de l'iceberg, mais quand même.

- Il y a tout le décodeur à programmer.
- alors, nous disposerons de deux outils : un générateur et un décodeur. Il sera très gai avec l'aide d'un enseignant intéressé de composer des scénarios en employant notamment les notions de référentiels et de variations, de les programmer et enfin de les tester avec les petits enfants.

J'insiste sur le mot composer. C'est bien plus difficile qu'on ne l'imagine. Ainsi nous avons eu beaucoup de mal les enseignants et moi. Nous n'avons d'ailleurs écrit un scénario que pour le problème des inversions.

Il ne suffit pas de récupérer des exercices dans des bouquins. L'ordinateur a des qualités propres qu'il faut employer, notamment une capacité de dialogue que n'offrent pas les livres.

c. les difficultés rencontrées.

- au niveau de l'analyse.

  Il y a bien sûr les problèmes de communication. D'une part un étudiant qui ne savait trop par quel bout commencer, qui piétinnait et s'exprimait plutôt mal. En face, deux membres du corps enseignant pleins de bonne volonté mais eux aussi assez désorientés. Nous avons rapidement défini les grandes lignes, la philosophie du projet.

C'était plus compliqué de les préciser. Nous ne connaissions chacun que notre discipline et nous renvoyions assez facilement les difficultés.

C'est l'intérêt d'un mémoire de confronter les étudiants à ces embûches.

Pour ma part, tout s'est arrangé quand le sujet s'est précisé.

Au niveau de la réalisation.

Contraintes techniques.

l'APPLE II possède 48 kram, pour mon projet, c'était un peu court. Pas parce que mon projet est plus gros qu'un autre, mais par sa nature.
D'abord, il me fallait une base de données en mémoire centrale et ensuite parce que le français est truffé d'exceptions, ce qui engendre des kilomètres d'instructions et sature la machine.
De plus, je découvrais le micro-ordinateur et j'ai perdu énormément de temps en manipulations que je prévoyais pas.

Difficultés de maintenance.

Les exceptions entravent les règles habituelles de design. (beaucoup penser avant de programmer). Les programmes seront un peu plus touffus, moins lisibles. Ils seront plus compliqués à modifier, la documentation sera plus longue à lire.

Par ailleurs, il m'a toujours semblé essentiel qu'un utilisateur puisse modifier aisément le vocabulaire. Mon choix est subjectif. Il n'a donc aucune raison d'être définitif.

De tels changements ne seront pas simples.

La base de données est saturée, si on ajoute un mot, il faut en ôter un autre. Il faut aussi entrer tout l'entourage sémantique de ce mot (quepeut-il suivre ou précéder, de quels verbes peut-il être sujet COD COI, à quelle classe appartient-il ?) et l'entourage syntaxique de ce mot (comment s'accorde-t-il ?)

Pour créer cette base de données, il m'a fallu écrire de longs programmes.
Je n'en parle pas dans le mémoire, ce n'est pas le but du travail mais cela m'a pris beaucoup de temps.

Je les ai écrit pour moi ces programmes, pour tester au plus vite mon générateur.
Ils ont peu de traitement d'erreur, ils ne sont pas "user friendly"

Avant de rendre ce mémoire, jem'appliquerai à les améliorer en ce sens.

Voilà, je ne rédige plus. J'espère que vous avez aimé ce projet, qu'il aura une suite. J'aimerais m'en occuper, c'était très gai.

POUR UN PROJET D'ENSEIGNEMENT
ASSISTE PAR ORDINATEUR SUR
L'ACCORD DU VERBE. (ANNEXES)

présenté par David Gouthière
promoteur Mr Claude Cherton

PLAN DES ANNEXES

* Les programmes de gestion de données ainsi que les listings du contenu complet de la base de données ne sont pas repris ici, ils sont trop volumineux !

Pour les obtenir :
- demander à Mr Cherton
- s'il les a perdus, lui demander les disquettes et exécuter les programmes d'impression.
- s'il les a perdues aussi, demander mon adresse au secrétariat de l'Institut et me contacter.

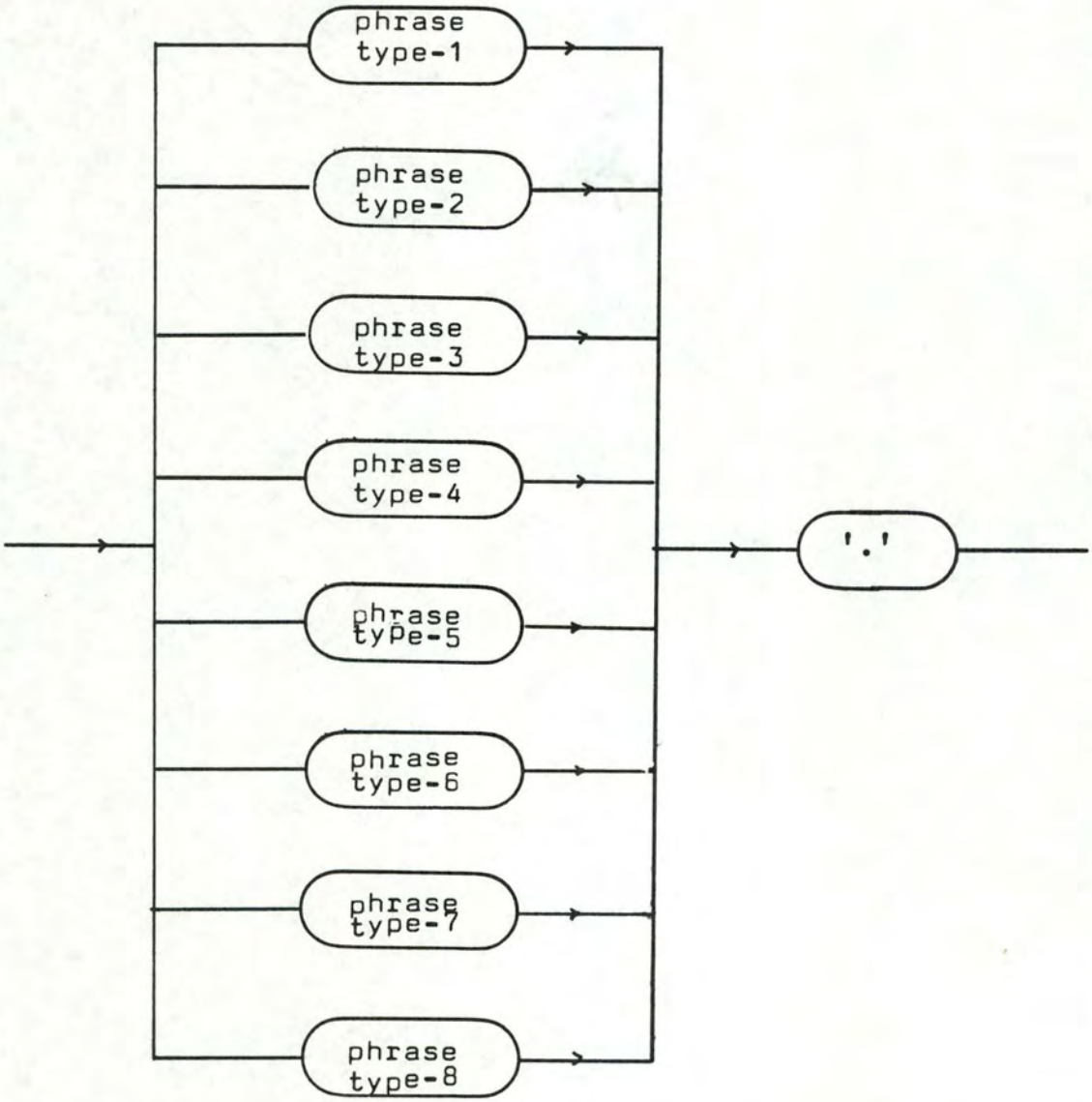
ANNEXE A.I

DIAGRAMMES SYNTAXIQUES, COMMENTAIRES, NOMENCLATURE

DIAGRAMMES SYNTAXIQUES ,COMMENTAIRES ,NOMENCLATURE .

Notation : Le signe "!" renseigne un élément que les diagrammes ne représentent pas ,mais dont il faut tenir compte.

phrase

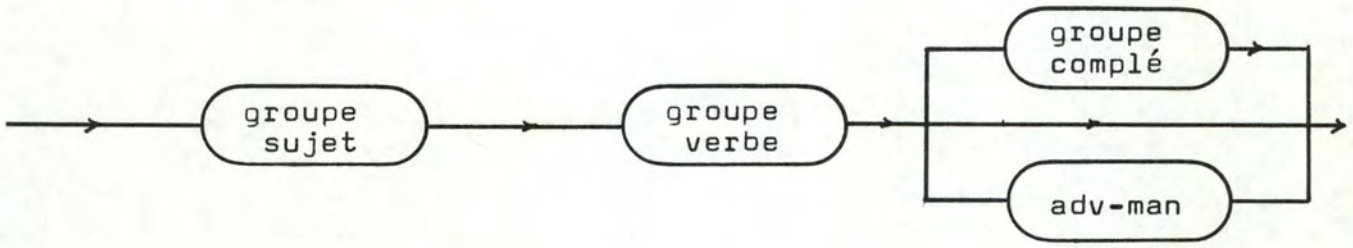


Ce diagramme montre qu'une phrase peut se définir de huit manières distinctes dans mon projet.

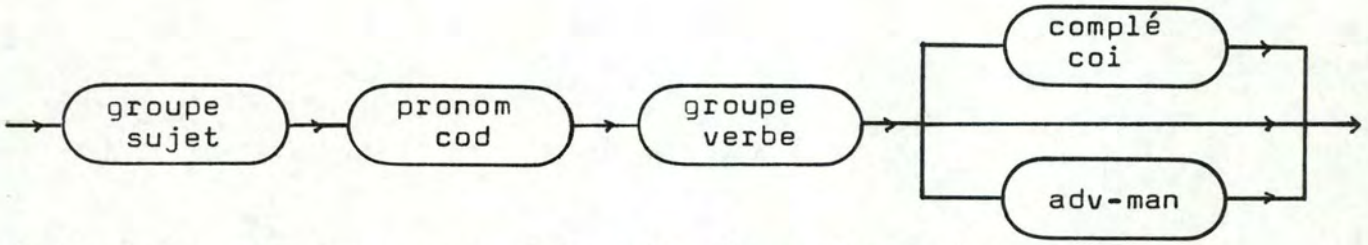
Il indique que la phrase se terminera par un point.

! Il ne précise pas que la première lettre du premier mot doit être une majuscule.

phrase-type-1

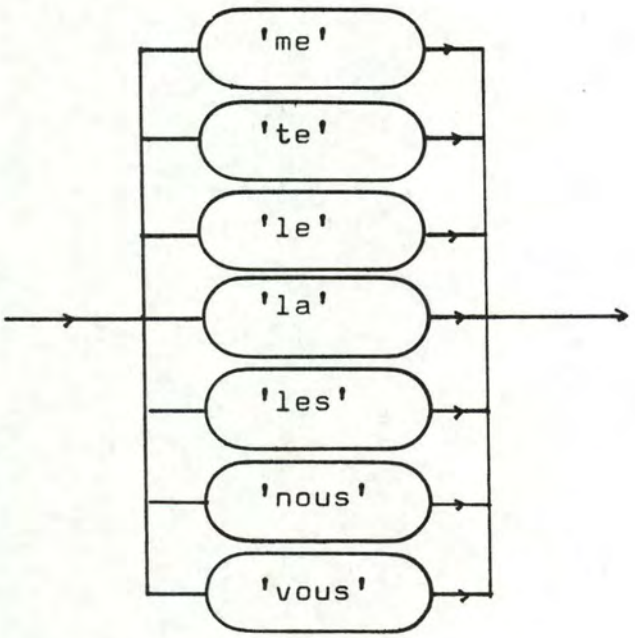


phrase-type-2



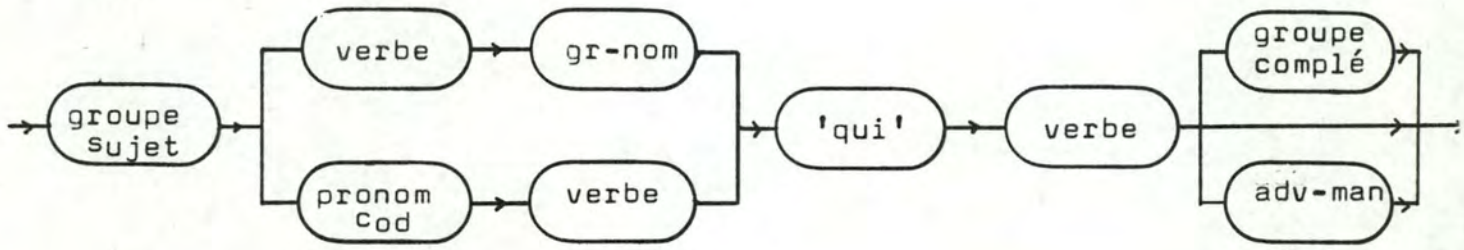
! Chaque verbe du groupe-verbe doit nécessairement accepter un complément d'objet direct.

pronom-cod



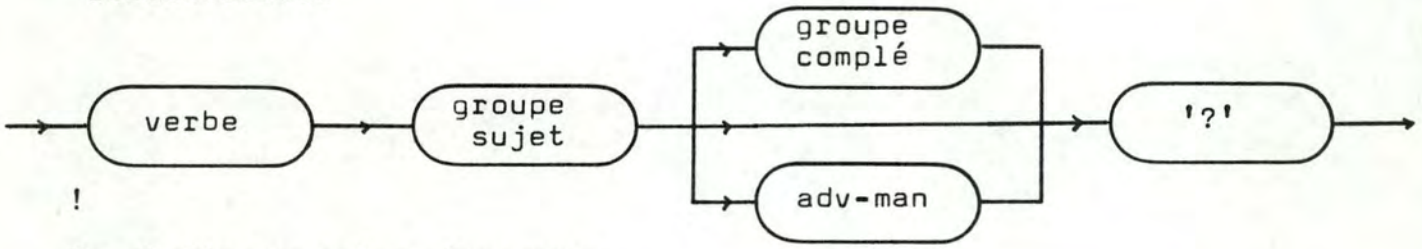
! Lorsque la dernière lettre du pronom et la première lettre du mot qui suit le pronom sont des voyelles, la dernière lettre du pronom est remplacée par une apostrophe ,le blanc qui sépare ces deux mots sera supprimé.

phrase-type-3



! Le premier verbe doit admettre un complément d'objet direct.

phrase-type-4



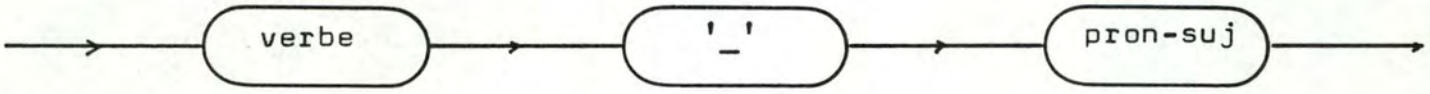
!

On distingue selon deux axes

- Le sujet est un pronom personnel ou un nom propre / commun
- Le temps du verbe est simple ou composé

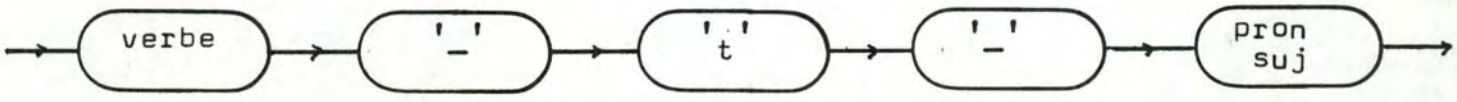
a. Le sujet est un pronom personnel.

1. Le temps est simple.

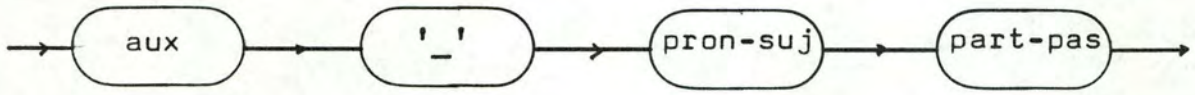


! Si le pronom sujet est "il,elle,on,ils,elles" et si le verbe ne se termine pas par "t"

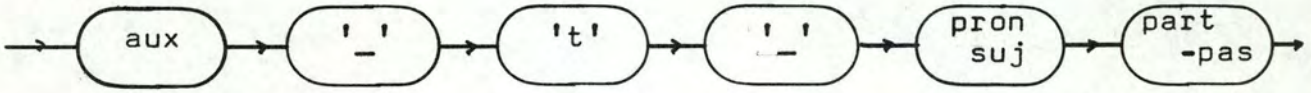
alors,on aura



2. Le temps est composé.

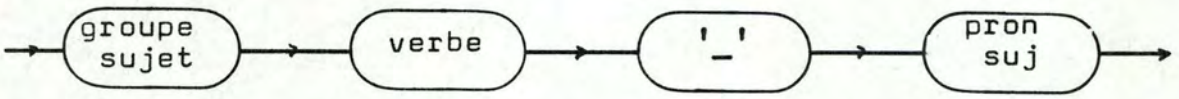


! Si le pronom sujet est "il,elle,on,ils,elles" et si l'auxiliaire du verbe ne se termine pas par "t" , on aura



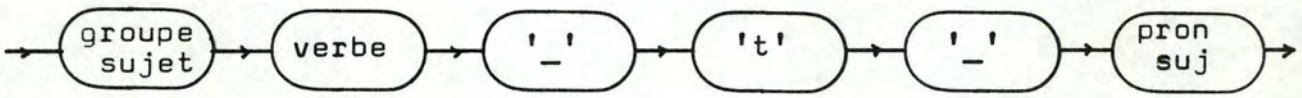
b. Le sujet est un nom propre ou un nom commun.

1. Le temps est simple.



! Le pronom sujet sera toujours "il,elle,ils,elles" .

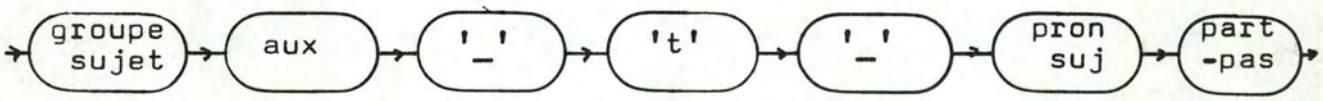
! Si de plus,le verbe ne se termine pas par la lettre "t" on aura



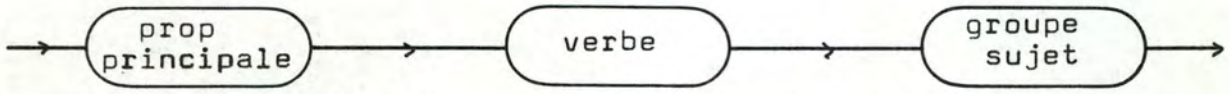
2. Le temps est composé

! Le pronom sujet sera nécessairement "il,elle,ils,elles".

! Si l'auxiliaire du verbe ne se termine pas par la lettre "t"

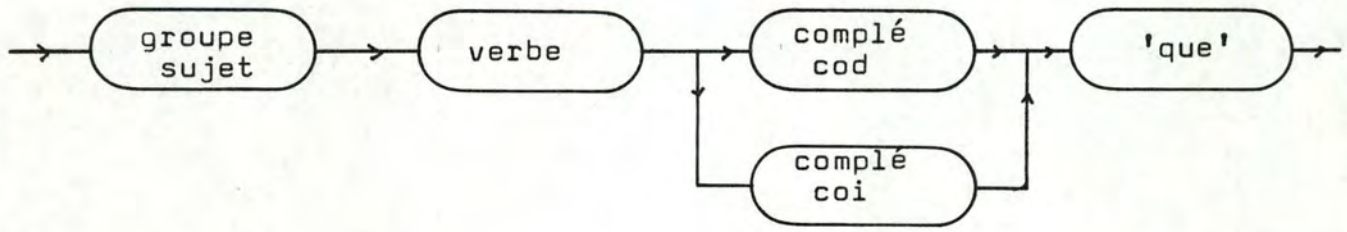


phrase-type-5

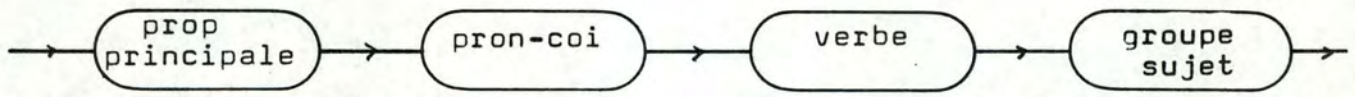


! Le sujet sera nécessairement un nom propre ou un nom commun.

prop principale

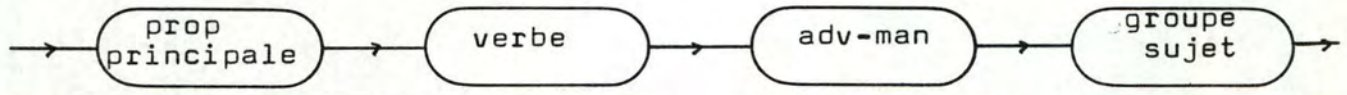


phrase-type-6



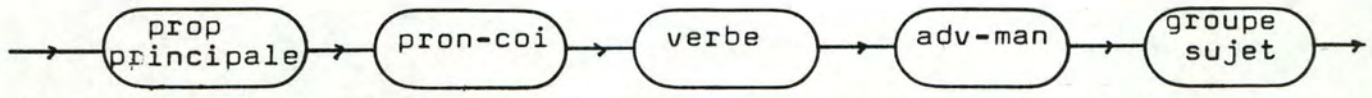
! Le sujet sera nécessairement un nom propre ou un nom commun.

phrase-type-7



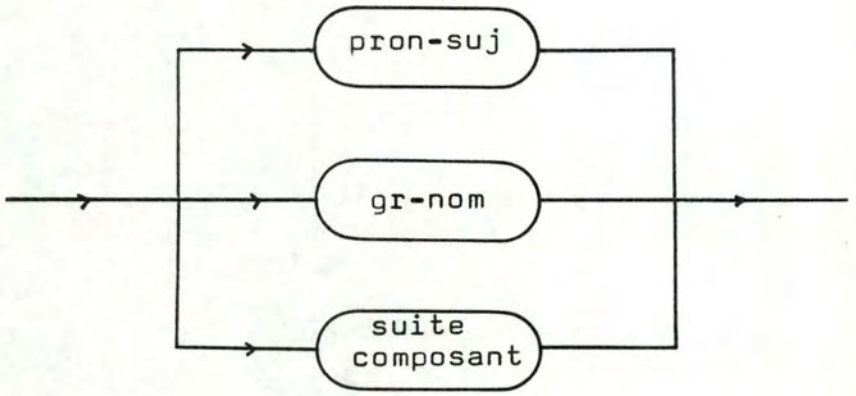
! Le sujet sera nécessairement un nom propre ou un nom commun.

phrase-type-8



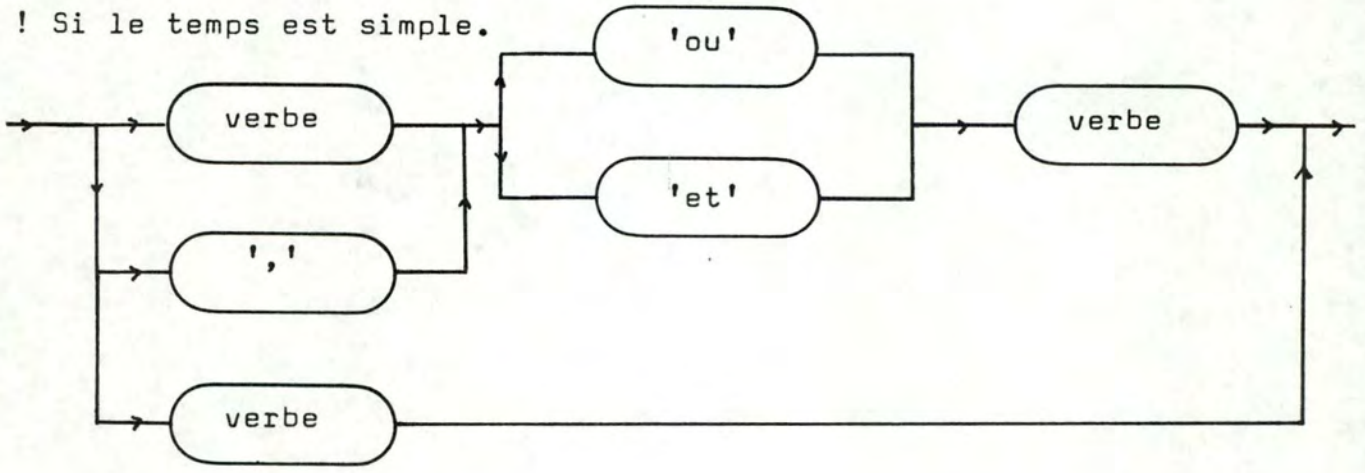
! Le sujet sera nécessairement un nom propre ou un nom commun.

groupe-sujet

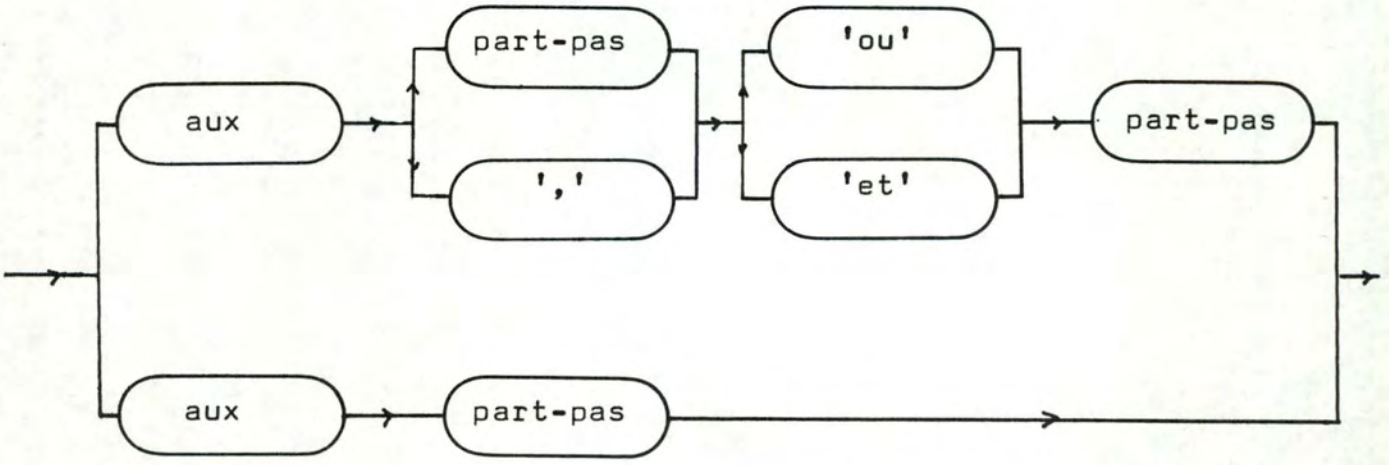


groupe-verbe

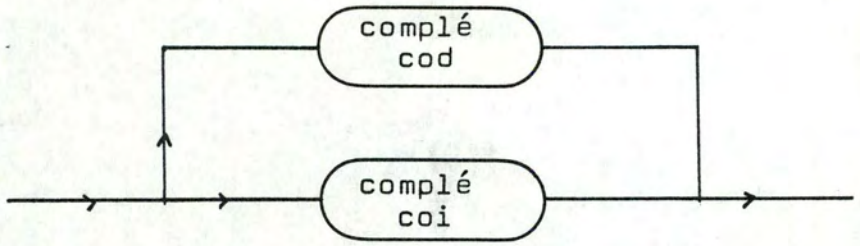
! Si le temps est simple.



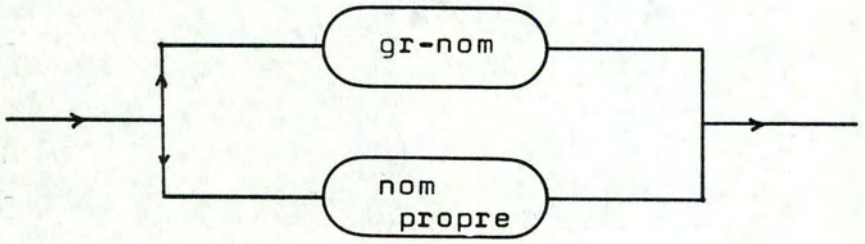
! Si le temps est composé.



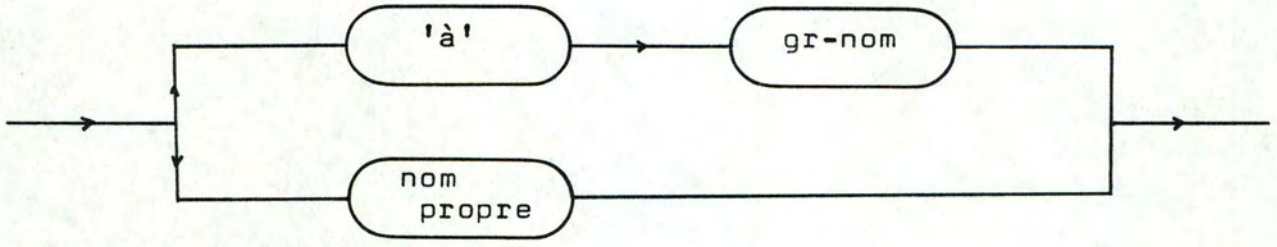
groupe-complé



complé-cod



complé-coi

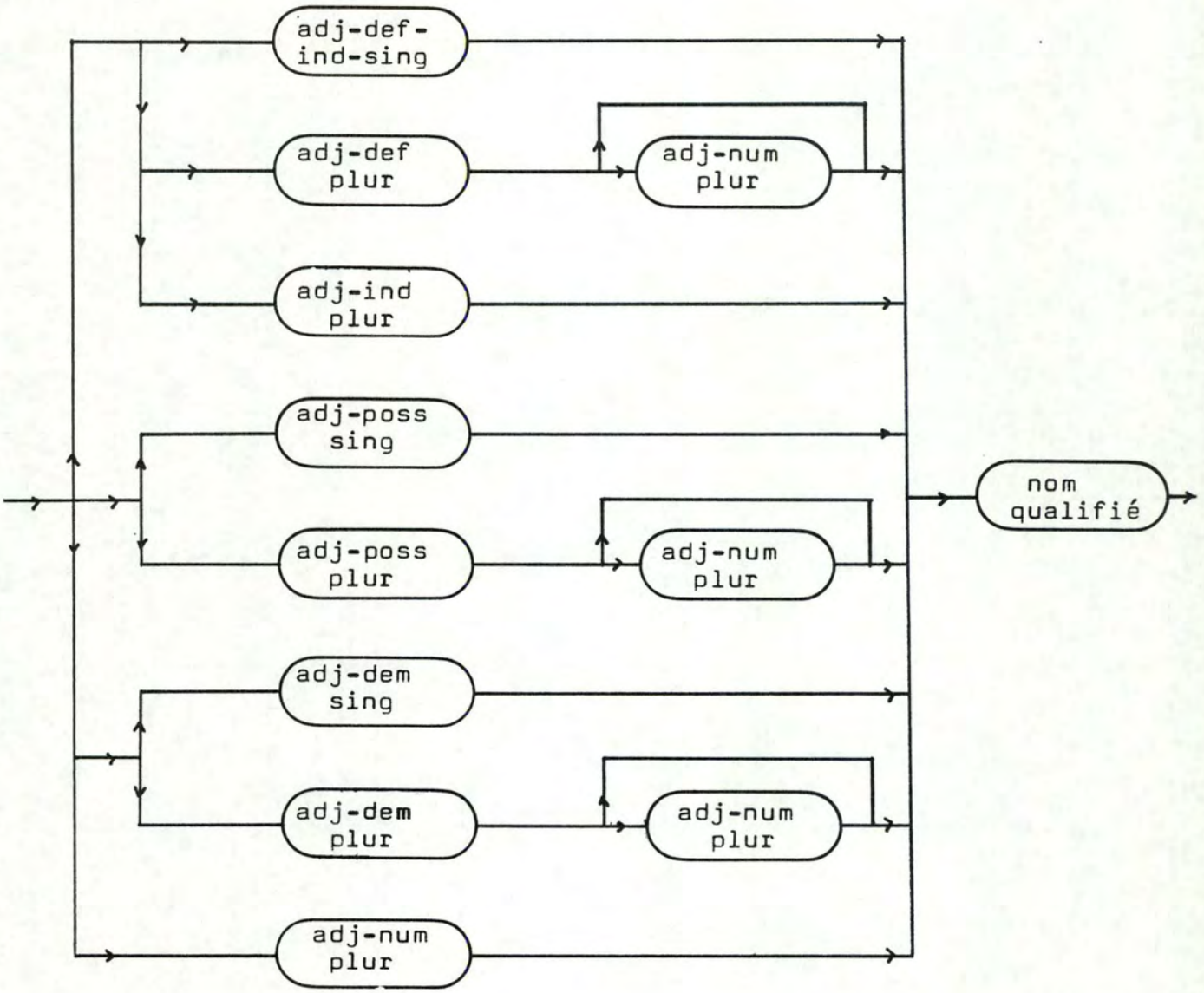


! La préposition "à" deviendra

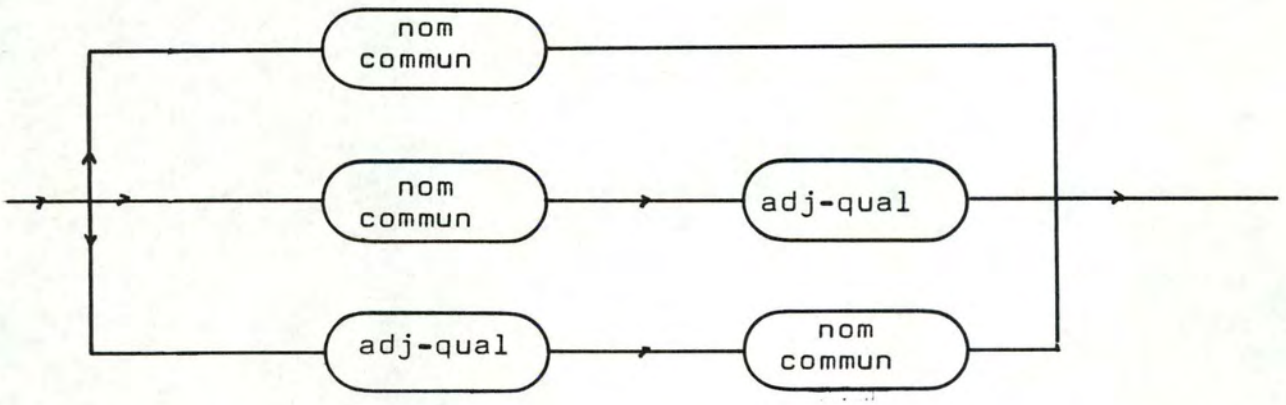
- "au" si le gr-nom commence par "le"
- "aux" si le gr-nom commence par "les"

dans les deux cas ,le premier mot de gr-nom est supprimé.

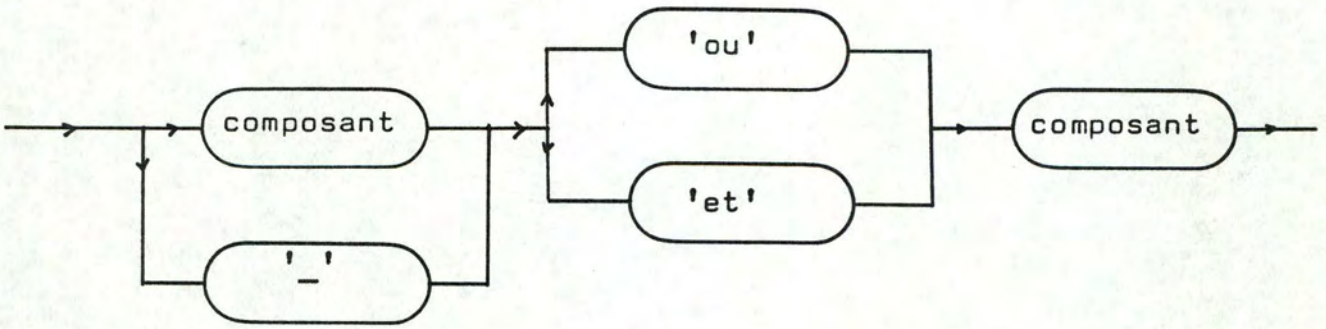
gr-nom



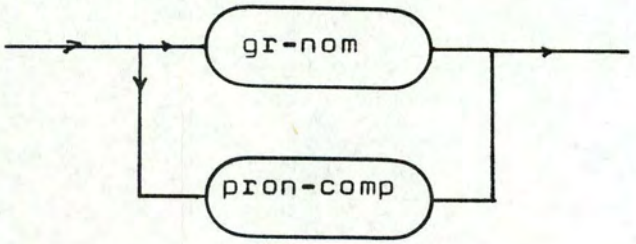
nom-qualifié



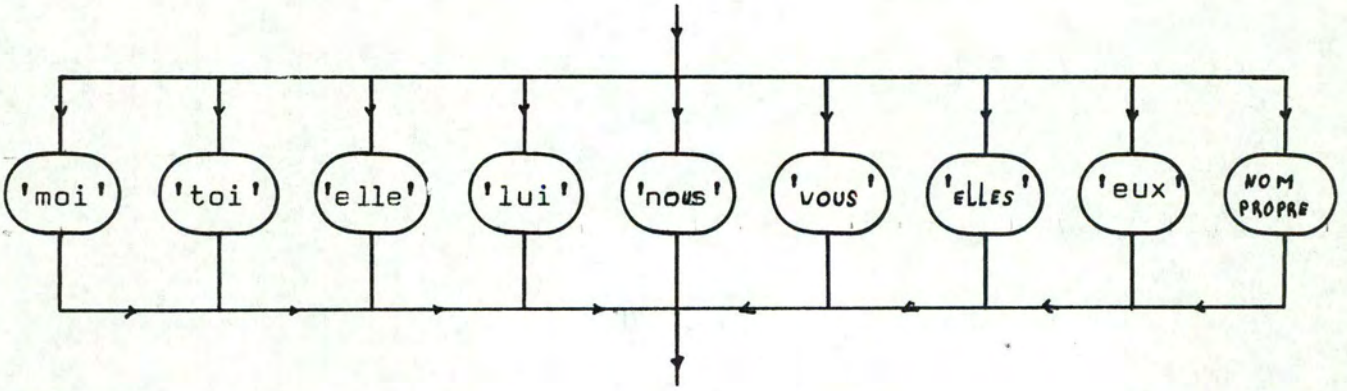
suite-composant



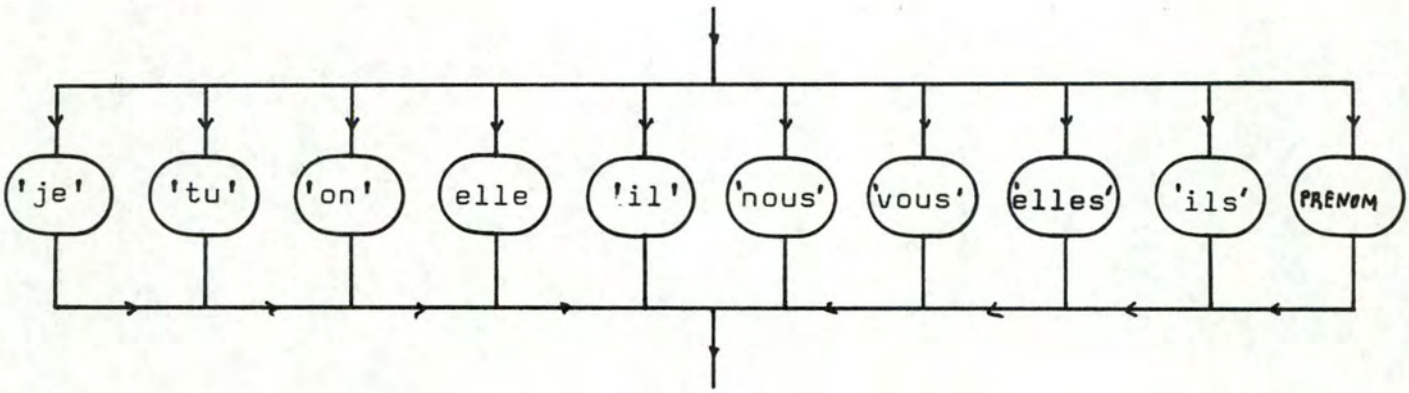
composant



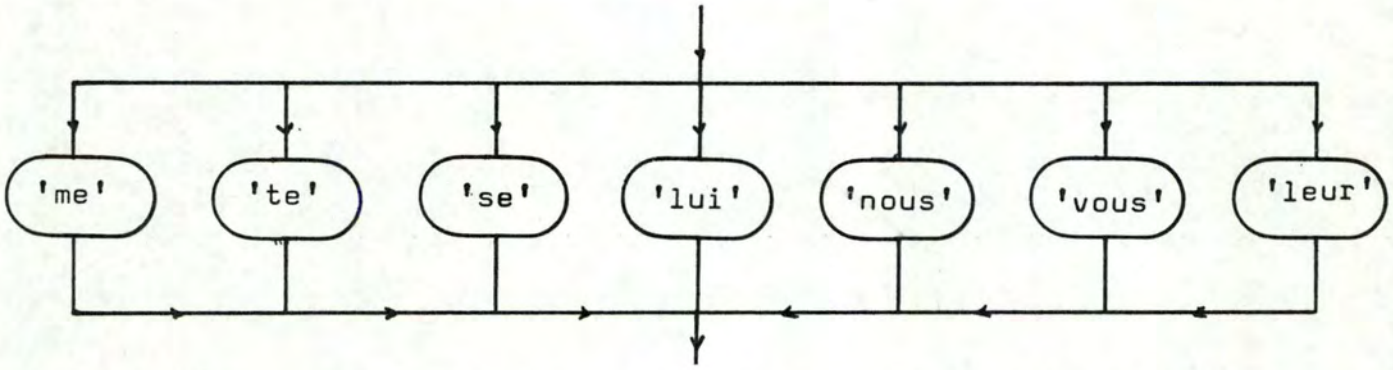
pron-comp



pron-suj

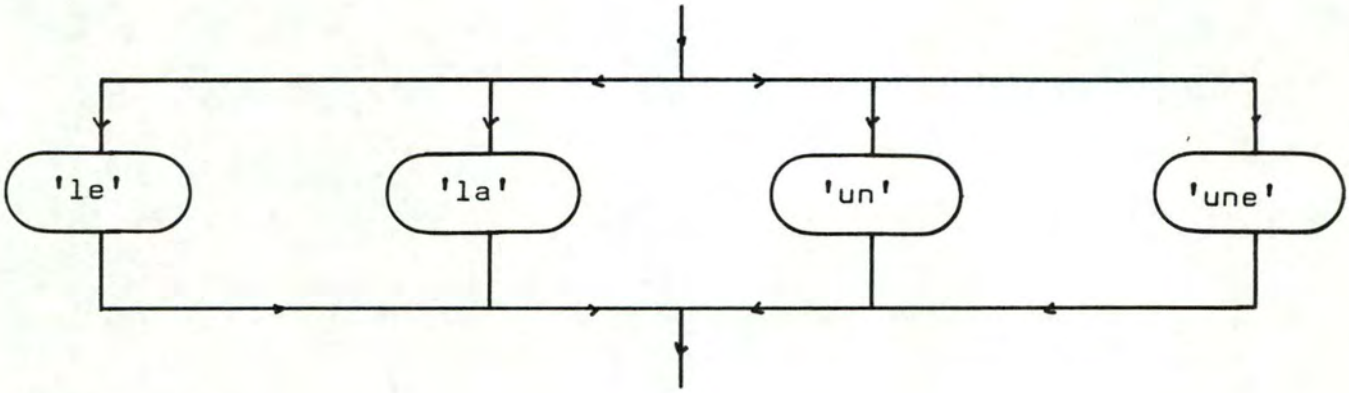


pron-coi

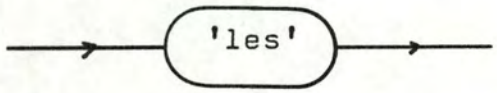


! Quand la première lettre du mot qui suit un pronom coi est une voyelle et que le pronom coi se termine par une voyelle, cette dernière voyelle est remplacée par une apostrophe, le blanc qui sépare les deux mots est supprimé.

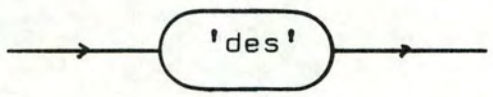
adj-def-ind-sing



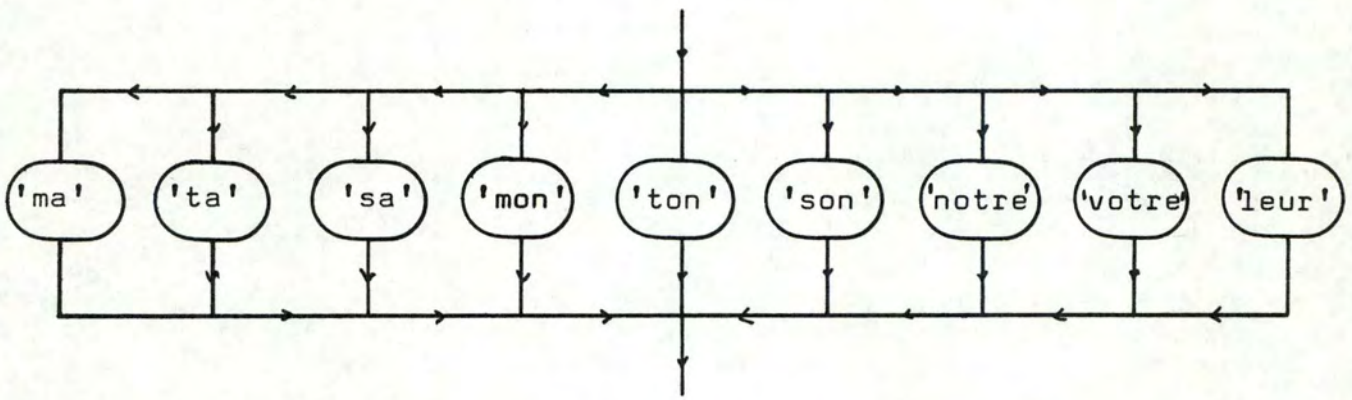
adj-def-plur



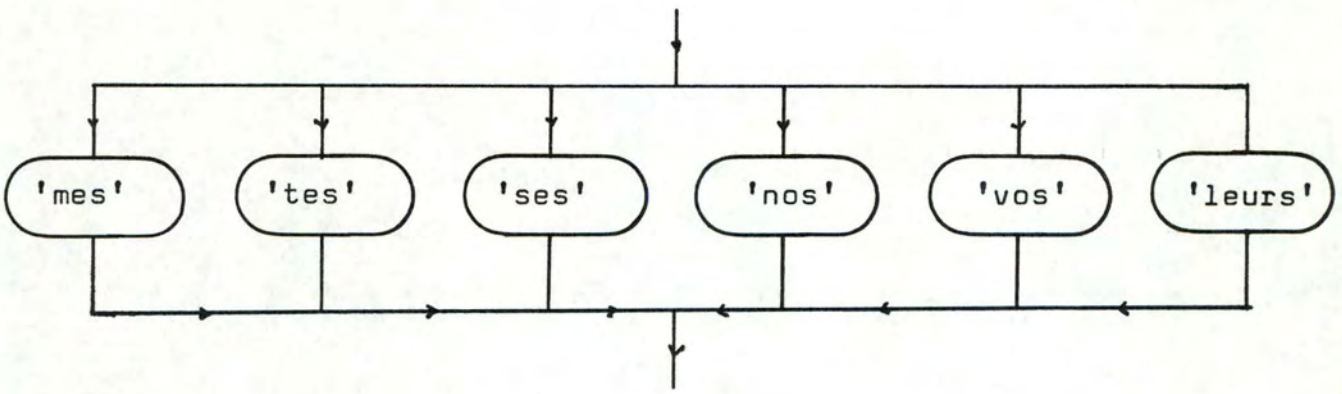
adj-ind-plur



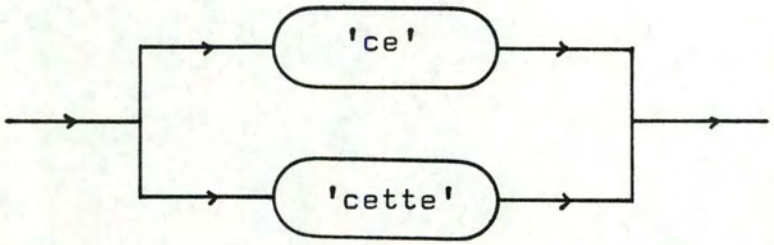
adj-poss-sing



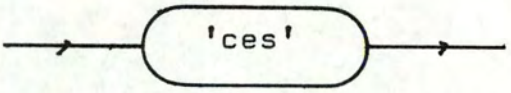
adj-poss-plur



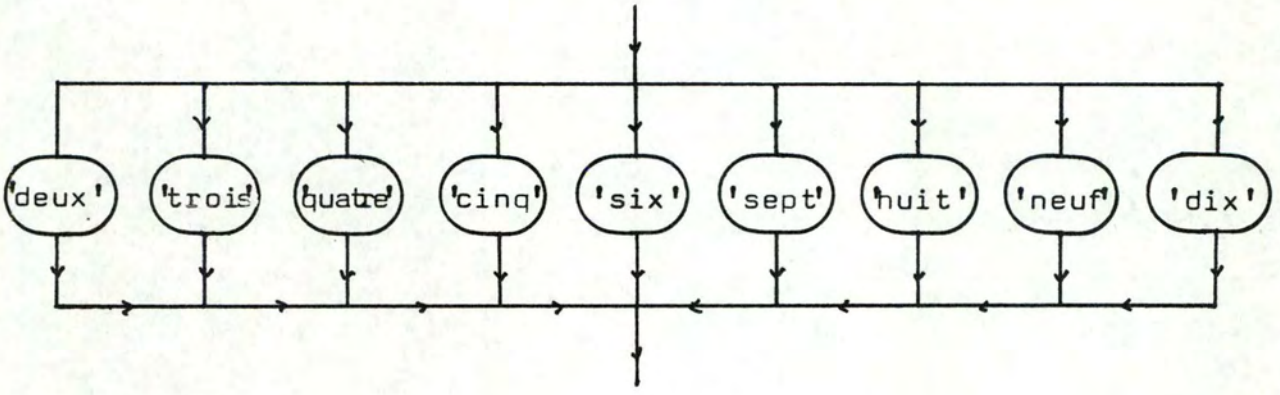
adj-dem-sing



adj-dem-plur



adj-num-plur



Nomenclature par ordre alphabétique,

```
adj-def-ind-sing  (adjectif défini ou indéfini singulier);
adj-def-plur  (adjectif défini pluriel);
adj-dem-plur  (adjectif démonstratif pluriel);
adj-dem-sing  (adjectif démonstratif singulier);
adj-ind-plur  (adjectif indéfini pluriel);
adj-num-plur  (adjectif numéral pluriel);
adj-poss-plur  (adjectif possessif pluriel);
adj-poss-sing  (adjectif possessif singulier);
adj-qual  (adjectif qualificatif);
adv-man  (adverbe de manière);
aux  (auxiliaire du verbe aux temps composés);
complé-cod  (complément d'objet direct);
complé-coi  (complément d'objet indirect);
composant  (composant d'un sujet composé);
groupe-complé  (groupe complément);
gr-nom  (groupe nom);
groupe-sujet ;
groupe-verbe ;
nom propre  ;
phrase  ;
phrase-type-1  ( premier type de phrase);
phrase-type-2  ;
phrase-type-3  ;
phrase-type-4  ;
phrase-type-5  ;
phrase-type-6  ;
phrase-type-7  ;
phrase-type-8  ;
part-pas  (participe passé du verbe);
pron-comp  (pronom d'un sujet composé);
pronom-cod  (pronom complément d'objet direct);
pronom-coi  (pronom complément d'objet indirect);
pron-suj  (pronom personnel sujet);
prop-principale  (proposition principale);
suite-composant  (sujet composé);
verbe  ;
```

Ce qui se trouve entre parenthèse n'est pas une définition mais simplement une aide pour déchiffrer les identificateurs.

ANNEXE A.II

SPECIFICATIONS DU PROGRAMME 'GENERATION'

. TYPES DE DONNEES

. VARIABLES

. PROCEDURES

SPECIFICATIONS DES TYPES DE DONNEES DU PROGRAMME 'GENERATION'

a. Les constantes
b. Les types propres à la sémantique
c. Les types propres à la syntaxe

La majorité des types de données présentés sont également employés dans la gestion de la base de données au niveau syntaxique (programme 'gesdictio') et au niveau sémantique (programmes'gersema' et 'term') .

Passons les en détail.

a. Voyons d'abord les constantes

```
CONST     NBCHAR=2000;
          NBMOT=290;
          NBMNP=17;
          NBFNP=17;
          NBVE=21;
          NBVIS=7;
          NBVS=16;
          NBNC=155;
          NBAQ=30;
          NBAM=30;
          NCH = 2000;
          NTERM = 500;
```

NBCHAR est la longueur maximum du type de tableau TABC2000 ;
NBMOT est la longueur maximum du type de tableau TABD290 ;
NBMNP est la longueur maximum du type de tableau TABIMASCNP ;
NBFNP est la longueur maximum du type de tableau TABIFEMNP ;
NBVE est la longueur maximum du type de tableau TABEVERBE ;
NBVIS est la longueur maximum du type de tableau TABISVERBE ;
NBVS est la longueur maximum du type de tableau TABSVERBE ;
NBNC est la longueur maximum du type de tableau TABNCOMMUN ;
NBAQ est la longueur maximum du type de tableau TABADJQUAL ;
NBAM est la longueur maximum du type de tableau TABADVERBE ;
NCH est la longueur maximum du type de tableau TABTERM ;
NTERM est la longueur maximum du type de tableau TABINDIR ;

b. Les types propres à la sémantique

. SEMADJDESC

```
SEMADJDESC = PACKED RECORD
                NBCL:1..3;
                LCLAS:PACKED ARRAY[1..3]OF INTEGER
             END;*)
```

Ce type d'enregistrement donne pour chaque **adjectif**

NBCL le nombre de classes d'adjectifs auxquelles appartient l'adjectif concerné (au plus trois)

LCLASS la liste des numéros de classes par ordre croissant ;

. VALCLAS

```
VALCLAS=PACKED RECORD
                  LISTEMOT:ARRAY[1..20] OF INTEGER;
                  LONGLIST:INTEGER
               END;
```

Ce type d'enregistrement donne pour chaque classe de noms

LONGLIST le nombre de noms que possède cette classe

LISTEMOT la liste des numéros d'ordre alphabétique de ces mots triés parmi l'ensemble des radicaux des noms communs ;

. FONC

```
FONC = (SUJ,COD,COIA,ADV);
```

Ensemble des rôles que peuvent jouer les groupes noms et les adverbes dans une phrase

SUJ sujet

COD complément d'ojet direct

COIA complément d'objet indirect introduit par la préposition à

ADV adverbe de manière ;

. SEMNCDESC

```
SEMNCDESC = PACKED RECORD

              GENERER:PACKED RECORD
                        NBADAV:INTEGER;
                        NBADAP:INTEGER;
                        PTADAV:PACKED ARRAY[1..8]OF INTEGER;
                        PTADAP:PACKED ARRAY[1..8]OF INTEGER
                      END;

              RECEVOIR:PACKED RECORD
                        LCLASSE:PACKED ARRAY[1..5]OF BOOLEAN
                      END;

              APPARTIENT:PACKED RECORD
                           NBCL:0..3;
                           LCLAS:PACKED ARRAY[1..3]OF INTEGER
                         END

            END;
```

Cet enregistrement se compose de trois parties,

GENERER contient de l'information sur les noms communs nécessaire à générer des groupes noms

- NBADAV nombre d'adjectifs que l'on a attribué à un nom pour le précéder
- NBADAP nombre d'adjectifs que l'on a attribué à un nom pour le suivre
- PTADAV liste des numéros des adjectifs 'avant' triés par ordre alphabétique des radicaux d'adjectifs
- PTADAP liste des numéros des adjectifs 'après' triés par ordre alphabétique des radicaux d'adjectifs ;

<u>petit exemple</u> : soit le mot 'poisson'

```
nbadav:3                    4  gros
nbadap:2                   13  petit
ptadav:4 13 33             25  rouge
ptadap:25 54               33  vieux
                           54  vert
```

Nous pourrions générer un 'petit poisson'
ou un 'poisson rouge' .

RECEVOIR contient de l'information sur les noms communs nécessaire pour décoder des phrases

LCLASSE rappelons-nous que les adjectifs sont répartis en cinq classes numérotées 1..5 ;
LCLASSE est un tableau de booleens à 5 entrées, une entrée par classe.
$LCLASSE_i$ = vrai si le nom commun accepte un adjectif de la classe i
faux sinon .

APPARTIENT ce type d'enregistrement donne pour chaque nom

NBCL nombre de classes noms auxquelles appartient ce nom
(trois au plus)
LCLAS liste des numéros de classes triés par ordre croissant.

. SEMVBDESC

```
SEMVBDESC = PACKED RECORD

                CODEXISTE:BOOLEAN;

                COIAEXISTE:BOOLEAN;


                RECEVVB:PACKED RECORD

                  SUJOK:PACKED RECORD
                     SNBCLAS:INTEGER;
                     STAB:PACKED ARRAY[1..25]OF BOOLEAN
                    END;

                  CODOK:PACKED RECORD
                     CODNBCLAS:INTEGER;
                     CODTAB:PACKED ARRAY[1..25]OF BOOLEAN
                    END;

                  COIAOK:PACKED RECORD
                     COIANBCLAS:INTEGER;
                     COIATAB:PACKED ARRAY[1..25]OF BOOLEAN
                    END

                END;


                GENERVB:PACKED RECORD

                  LISTSUJ:PACKED ARRAY[1..20]OF INTEGER;
                  NBSUJ:INTEGER;
                  ADVTAB:PACKED ARRAY[1..10]OF INTEGER;
                  NBADV:INTEGER;

                  COD:PACKED RECORD
                     CODLIST:PACKED ARRAY[1..20]OF INTEGER;
                     NBCOD:INTEGER
                    END;

                  COIA:PACKED RECORD
                     COIALIST:PACKED ARRAY[1..20]OF INTEGER;
                     NBCOIA:INTEGER
                    END;

                END;


              END;
```

CODEXISTE vrai si le verbe concerné admet un complément d'objet direct dans le cadre de ce projet ;

COIAEXISTE vrai si le verbe concerné admet un complément d'objet indirect dans le cadre de ce projet ;

RECEVVB contient de l'information sur les verbes nécessaire à décoder les phrases d'un enfant

SUJOK

SNBCLAS nombre de classes de noms dont je juge que les éléments peuvent être sujet du verbe concerné dans l'esprit de l'enfant

STAB tableau de booleens,à chaque entrée correspond une classe de noms; STAB . est vrai si j'estime que les éléments de la classe i peuvent être sujet du verbe dans l'imagination d'un petit élève

CODOK

CODNBCLAS nombre de classes de noms dont je juge que les éléments peuvent être complément d'objet direct du verbe concerné dans l'esprit de l'enfant

CODTAB tableau de booleens,à chaque entrée correspond une classe de noms; CODTAB i est vrai si j'estime que les éléments de la classe i peuvent être complément d'objet direct du verbe dans l'esprit de l'enfant

COIAOK

COIANBCLAS nombre de classes noms dont je juge que les éléments peuvent être complément d'objet indirect du verbe concerné dans l'esprit de l'enfant

COIATAB tableau de booleens,à chaque entrée correspond une classe de noms; COIATAB i est vrai si j'estime que les élémentsde la classe i peuvent être complément d'objet indirect du verbe dans l'esprit de l'enfant ;

GENERVB contient de l'information nécessaire à la génération d'une phrase

LISTSUJ liste des classes de mots dont je juge que les élément peuvent jouer le rôle de sujet

NBSUJ nombre d'éléments de LISTSUJ

ADVTAB liste des numéros des adverbes que je juge pouvoir accompagner le verbe

NBADV nombre d'éléments de ADVTAB

COD

CODLIST liste des classes de mots dont je juge que les éléments peuvent jouer le rôle de cod

NBCOD nombre d'éléments de CODLIST

COIA

COIALIST	liste des classes de mots dont je juge que les éléments peuvent jouer le rôle de coi
NBCOI	nombre d'éléments de COIALIST ;

c. types propres à la syntaxe

. conjugaison
. table de noms, de verbes, d'adjectifs, d'adverbes, de noms propres

1. conjugaison

```
TABTERM = PACKED ARRAY [1..NCH] OF CHAR;

TABINDIR = PACKED ARRAY [1..NTERM] OF INTEGER;

TABSTRING = PACKED ARRAY [1..6] OF STRING8;

GROUPE = PACKED RECORD
              PRESENT:TABSTRING;
              IMPARFAIT:TABSTRING;
              PASSESIMPLE:TABSTRING;
              FUTURSIMPLE:TABSTRING;
              PARTPASSE:STRING4
         END;

TERMCONJ = FILE OF GROUPE;
```

. TABTERM

Type de tableaux qui contient bout à bout sans séparation les terminaisons de chaque conjugaison considérée.

Les terminaisons sont triées

- par ordre croissant des numéros de groupe tels qu'ils a apparaissent dans le guide BESCHERELLE*
- par ordre de temps au sein d'un groupe (présent,imparfait,passé simple,futur simple,participe passé
- par ordre de personne au sein d'untemps (1ère,2ème,3ème singulier,1ère,2ème,3ème pluriel)

Sa longueur est la somme des longueurs de toutes les terminaisons en n'oubliant pas de compter le séparateur.

. TABINDIR

Ce type de tableau contient les adresses du début des terminaisons dans un tableau de type TABTERM

Sa longueur qui est le nombre de terminaisons vaut :

- nombre de groupes X nombre de terminaisons par groupe

où nombre de terminaisons par groupe égal

nombre de personnes X nombre de temps + une (participe passé)

soit 6 X 4 + 1 = 25

. TABSTRING

C'est un type de tableau de strings associé à un temps.

Le 1er contient la terminaison de la 1ere personne singulier
.. 2è ..... 2è .....
.. 3è ..... 3è .....
.. 4è ..... 1ère ..... pluriel
.. 5è ..... 2è .....
.. 6è ..... 3è .....

. GROUPE

C'est un type d'enregistrement qui contient les terminaisons des 4 temps (présent,imparfait,passé simple,futur simple) et du participe passé d'un groupe BESCHERELLE.

. TERMCONJ

C'est un type de fichier de GROUPE trié par ordre de numéro de groupe

2. tables des noms,adjectifs,verbes,adverbes,noms propres

```
NATURE=(FNPROPRE,MNPROPRE,NCOM,EVERBE,ISVERBE,SVERBE,ADJQ,ADVMAN);
```

. NATURE

C'est l'ensemble des natures des différents mots enregistrés dans la base de données.

FNPROPRE : nom propre féminin
MNPROPRE : nom propre masculin
NCOM : nom commun
EVERBE : verbe appartenant à un groupe EISS E
(retenez que ce sont les verbes en er)
ISVERBE : appartenant à un groupe EISS IS
(retenez que ce sont les verbes qui font issant
au participe présent ex finissant)
SVERBE : verbe qui n'est ni en E, ni en IS
ADJQ : adjectif qualificatif
ADVMAN : adverbe de manière

. SUFFIXE

```
SUFFIXE=PACKED RECORD
                  FSY:BOOLEAN;
                  FPY:BOOLEAN;
                  MSY:BOOLEAN;
                  MPY:BOOLEAN;
                  FS:STRING8;
                  FP:STRING8;
                  MS:STRING8;
                  MP:STRING8;
                END;
```

Type d'enregistrement composé qui contient

FSY booleen, vrai si le mot admet un féminin singulier
faux sinon
FPY booleen, vrai si le mot admet un féminin pluriel
faux sinon
MSY booleen, vrai si le mot admet un masculin singulier
faux sinon
MPY booleen, vrai si le mot admet un masculin pluriel
faux sinon
FS si FSY est vrai, c'est la terminaison fém.sing du mot
faux, indéfini
FP si FPY est vrai, c'est la terminaison fém.plur du mot
faux, indéfini
MS si MSY est vrai, c'est la terminaison masc.sing du mot
faux, indéfini
MP si MPY est vrai, c'est la terminaison masc.plur du mot
faux,indéfini

. SUFFVB

```
SUFFVB=PACKED RECORD
                INF:STRING5;
                CONJ:INTEGER
              END;
```

Type d'enregistrement qui contient

INF la terminaison infinitive d'un verbe
CONJ son numéro d'ordre de conjugaison
attention ! ce n'est pas son numéro de groupe BESCHERELLE
c'est le numéro d'ordre de son groupe BESCHERELLE

. STXVBDESC

```
STXVBDESC = PACKED RECORD
                  TERMVB:SUFFVB;
                  ADIND:INTEGER
                END;
```

Type d'enregistrement qui contient

TERMVB la description de la partie variable (SUFFVB) d'un verbe
ADIND le numéro d'ordre alphabétique du radical d'un verbe
parmi l'ensemble des radicaux de tous les mots du
dictionnaire

. STXADJDESC

```
STXADJDESC = PACKED RECORD
                   TERMADJ:SUFFIXE;
                   ADIND:INTEGER
                 END;
```

Typè d'enregistrement qui contient

TERMADJ la description de la partie variable (SUFFIXE) d'un adjectif qualificatif

ADIND le numéro d'ordre alphabétique du radical d'un adjectif parmi l'ensemble des radicaux de tous les mots du dictionnaire

. STXNCDESC

```
STXNCDESC = PACKED RECORD
                    CARACT:PACKED ARRAY[1..6] OF BOOLEAN;
                    TERM:SUFFIXE
                END;
```

Type d'enregistrement qui contient

CARACT les caractéristiques du nom commun. C'est un tableau de booleens qui répond successivement à 6 questions

1 - ce nom commun a-t-il une forme fém.sing ? (**vrai**/faux)
2 - ..... fém.plur .....
3 - ..... mas.sing .....
4 - ..... mas.plur .....
5 - ..... admet-il un adj.qual avant? .....
6 - ..... admet-il un adj.qual après? .....

TERM le descripteur de la partie variable du nom commun

. STXNCOM

```
STXNCOM = PACKED RECORD
                STX:STXNCDESC;
                ADIND:INTEGER
            END;
```

Type d'enregistrement qui contient

STX le descripteur de la partie variable et les caractéristiques du nom commun (STXNCDESC)

ADIND le numéro d'ordre alphabétique du radical du nom commun parmi l'ensemble des radicaux de tous les mots

. DESCRIPTMOT

```
DESCRIPTMOT = PACKED RECORD
                    NAT:NATURE;
                    NUMPARNAT:INTEGER;
                    ADRESSE:INTEGER
                END;
```

Ce type d'enregistrement contient

NAT La nature du mot décrit Cf NATURE
NUMPARNAT le numéro d'ordre alphabétique du radical du mot parmi l'ensemble des mots de même nature que lui
ADRESSE Adresse du debut du radical de ce mot dans le tableau de type TAB2000, avant d'en savoir plus sur ce dernier type,retenez que c'est un type de tableau où s'enfilent bout à bout tous les radicaux par ordre alphabétique

. WORD

```
WORD = PACKED RECORD
                ORTHO:STRING20;
                NAT:NATURE
           END;
```

Type d'enregistrement qui contient

ORTHO le radical d'un mot
NAT la nature du même mot

. TABC2000

```
TABC2000 = PACKED ARRAY[1..NBCHAR] OF CHAR;
```

Ce type de tableau enregistre bout à tous les radicaux des mots du vocabulaire

- triés par ordre alphabétique
- le caractère ' ' commence et fini ce type de tableau, et sépare chaque radical du suivant
- le dernier mot est bidon, c'est 'ZZ'

. TABD290

```
TABD290 = ARRAY[1..NBMOT] OF DESCRIPTMOT;
```

La ième entrée de ce type de tableau donne la description du ième mot du vocabulaire par ordre alphabétique

. TABINBFNP

```
TABINBFNP = ARRAY[1..NBFNP] OF INTEGER;
```

Type de tableaux des noms propres féminin
La ième entrée de ce type de tableau est un entier qui donne l'adresse dans un tableau de type TABD290 du descripteur du ième nom propre féminin par ordre alphabétique des noms propres féminins

. TABINBMNP

```
TABINBMNP = ARRAY[1..NBMNP] OF INTEGER;
```

Type de tableaux des noms propres masculins.
La ième entrée de ce type de tableau est un entier qui donne l'adresse dans un tableau de type TABD290 du descripteur du ième nom propre masculin par ordre alphabétique des noms propres masculins

. TABSNBNCOM

```
TABSNBNCOM = ARRAY[1..NBNC] OF STXNCOM;
```

Type de tableaux des descripteurs de noms communs.
La ième entrée de ce type de tableau est un descripteur STXNCOM du ième nom commun par ordre alphabétique des noms communs

. TABSENBV, TABSISNBV, TABSSNBV

```
TABSENBV = ARRAY[1..NBVE] OF STXVBDESC;

TABSISNBV = ARRAY[1..NBVIS] OF STXVBDESC;

TABSSNBV = ARRAY[1..NBVS] OF STXVBDESC;
```

Types de tableaux des descripteurs des verbes en e is s.
Le ième élément d'un tableau de ces types est un descripteur STXVBDESC du ième verbe e is s, par ordre alphabétique des verbes en e, is, s.
Les radicaux des verbes en e, is, s forment 3 classes distinctes.

. TABSNBADJQ

```
TABSNBADJQ = ARRAY[1..NBAQ] OF STXADJDESC;
```

Type de tableaux des descripteurs des adjectifs qualificatifs.
Le ième élément d'un tableau de ce type est un descripteur STXADJDESC du ième adjectif qualificatif par ordre alphabétique des adjectifs qualificatifs

. TABINBADV

```
TABINBADV = ARRAY[1..NBAM] OF INTEGER;
```

Type de tableaux des adverbes de manière.
Le ième élément d'un tableau de ce type est un entier qui donne l'adresse dans un tableau de type TABD290 du descripteur du ième adverbe de manière par ordre alphabétique des adverbes de manière

d. types de données propres au programme 'génération'

Bien entendu, les autres types de données sont aussi propres à la génération, mais ils décrivent surtout la structure de la base de données.

Voici maintenant les types de données dans lesquels j'enregistre les phrases que je construis.

La majorité des mots sont enregistrés dans des variables de type string, mais quelques associations de mots nécessitent des types de données plus structurés.

. GRNOM

```
GRNOM=PACKED RECORD
               DETERMINANT:PACKED RECORD
                             D1:STRING8;
                             D2:STRING8
                           END;
               ADAV:STRING20;
               NOM:STRING20;
               ADAP:STRING20;
               INDIC:PACKED ARRAY[1..5] OF INTEGER;
             END;
```

Type d'enregistrement en 6 parties.
Revoyez un peu les diagrammes syntaxiques qui décrivent un groupe nom pAI. Si,si j'insiste. Ne vous en faites pas, je vous attends.

...
Vous y êtes ? ok!

Il y a six boites

Dans la peemière, DETERMINANT.D1, il y aura le déterminant du nom s'il en possède un ou rien du tout.

au choix
- adjectif défini
- adjectif indéfini
- adjectif possessif
- adjectif démonstratif
- rien du tout

Dans la seconde, DETERMINANT.D2, il y aura un adjectif numéral (qui peut aussi jouer le rôle de determinant) ou rien du tout.

- adjectif numéral
- rien du tout

Dans la troisième, ADAV il y aura un adjectif qualificatif ou rien du tout.

- adjectif qualificatif
- rien du tout

Dans la quatrième, NOM il y aura un nom commun, toujours.

- nom commun

Dans la cinquième, ADAP il y aura un adjectif qualificatif ou ou rien du tout.

- adjectif qualificatif
- rien du tout

Dans la sixième, INDIC il y aura un indicateur (tableau de 5 entiers) qui me rappelle de quoi est fait le groupe nom.

INDIC [1] =	0	la 1ere boite est	vide
	1	.....	adjectif défini
	2	.....	adjectif indéfini
	3	.....	adjectif possessif
	4	.....	adjectif démonstratif
INDIC[2]=	0	.. 2ème	vide
	1	.....	adjectif numéral
INDIC[3]=	0	.. 3ème	vide
	1	.....	adjectif qualificatif
INDIC[4]=	0	.. 4ème	indéfinie
	1	.....	nom commun
INDIC[5]=	0	.. 5ème	vide
	1	.....	adjectif qualificatif

Deux remarques s'imposent à mes yeux.

- Pensez-vous que le groupe peut se composer du seul nom commun parce que 4 boites peuvent être vides ? Il n'en est rien.

  Il existe des contraintes qui me disent que 3 ou 4 boites doivent toujours être pleines ensemble.

  Vous les trouverez soit sous formes de diagrammes syntaxiques soit quand je vous présenterai l'algorithme qui construit le groupe nom.

- Les boites se présentent dans l'ordre des mots du groupe noms. C'est pour cela qu'il y a 2 boites adjectifs (une avant et une après le nom).

. SUITCOMPOSANT

```
SPRONOM=STRING20;

DESC=PACKED RECORD
        NC:CHAR;
        PC:CHAR;
        GC:CHAR;
        CASE TC:CHAR OF
          'P':(CP:SPRONOM);
          'G':(CG:GRNOM)
      END;

SUITCOMPOSANT=PACKED RECORD
                COMP:ARRAY[1..3] OF DESC;
                NBCOMP:INTEGER
              END;
```

SPRONOM C'est un type de variable qui contiendra un pronom personnel ou un prénom. (au point de vue de la syntaxe, prénom = 3ème pers.sing d'un pronom).

DESC Ce type de donnée recèle la description et la valeur d'un composant et d'une suite de composants (rappelez-vous les diagrammes syntaxiques).
ex. Hervé, sa grande soeur et moi

NC : 'S' si ce composant est singulier
'p' ..... pluriel

PC : '1' si ce composant est à la 1ère personne
'2' ..... 2ème .....
'3' ..... 3ème .....

GC : 'F' si ce composant est féminin
'M' ..... masculin

TC : 'P' si ce composant est un pronom personnel ou un prénom
'G' ..... groupe nom

SUITCOMPOSANT

NBCOMP : 2 si la suite à 2 composants
3 ..... 3 .....

. SUJETDESC

```
SUJETDESC=PACKED RECORD
              NOMBRE:CHAR;
              PERSONNE:CHAR;
              GENRE:CHAR;
              VAL:PACKED RECORD
                     CASE TYPSUJ:CHAR OF
                       'P':(PRONOM:SPRONOM);
                       'G':(GR:GRNOM);
                       'S':(ST:SUITCOMPOSANT)
                     END
            END;
```

Ce type de données décrit le sujet de la phrase.

NOMBRE : 'S' si le sujet est singulier
'P' ..... pluriel

PERSONNE : '1' si le sujet est à la 1ère personne
'2' ..... 2ème .....
'3' ..... 3ème .....

GENRE : 'P' si le sujet est un pronom **personnel** ou un prénom
VAL.PRONOM est la valeur du sujet

'G' si le sujet est un groupe nom
VAL.GR est la valeur du sujet

'S' si le sujet est une suite de composants
VAL.ST est la valeur du sujet.

SPECIFICATIONS DES VARIABLES DU PROGRAMME 'GENERATION'

Je vous présenterai successivement

- Les variables de la base de données communes à tous les types de mots
- Les variables spécifiques à la manipulation des noms propres féminins, des noms propres masculins, des noms communs, des verbes en E IS S, des adjectifs qualificatifs, des adverbes
- Les variables spécifiques à la génération

a. Les variables de la base de données communes à tous les types de mots

. LISTMOT : TABC2000;

Tableau de caractères. C'est lui qui contient bout à bout, séparés d'un caractère spécial '@', tous les radicaux du vocabulaire classés par ordre alphabétique. Le premier caractère est '@' ainsi que le dernier. Le dernier radical est bidon, c'est 'ZZ'.

. INDICE : TABD290;

C'est un tableau d'indirection.
INDICE [i] contient les informations qui permettent de connaitre :

- la nature
- l'adresse dans une table d'un descripteur où l'on des informations propres à la nature du ième mot
- l'adresse dans LISTMOT du ième radical du vocabulaire

. NBREMOT : integer;

Numéro du plus grand mot dans INDICE ('ZZ').

. NBRECAR : integer;

C'est la somme des caractères de tous les radicaux, plus ceux du mot bidon, plus le nombre de caractères '@'. C'est le plus grand indice de LISTMOT.

. FICHPREFIXE : file of WORD;

Fichier des radicaux des mots enregistrés du vocabulaire. Il est trié selon l'ordre alphabétique des radicaux.

b. Les variables spécifiques à la manipulation des noms propres féminins

. TFNPROP : TABINBFNP;

TFNPROP[i] est l'adresse dans INDICE du ième nom propre féminin par ordre alphabétique.

. NUMFNP : integer;

Numéro du plus grand nom propre féminin.

c. Les variables spécifiques à la manipulation des noms propres masculins

. TMNPROP : TABINBMNP;

TMNPROP[i] est l'adresse dans INDICE du ième nom propre masculin par ordre alphabétique.

. NUMMNP : integer;

Numéro du plus grand nom propre masculin..

d. Les variables spécifiques à la manipulation des noms communs

. TNCOM : TABSNBNCOM;

TNCOM[i] contient

- Les différentes formes de terminaisons du ième nom commun (SUFFIXE)
- Les caractéristiques de ce nom commun (CARACT)
- L'adresse dans INDICE du ième nom commun par ordre alphabétique.

. NUMNC : integer;

Numéro du plus grand nom commun.

. FSTXNCOM: file of STXNCDESC;

Fichier des terminaisons des noms communs et de leurs caractéristiques, il est trié par ordre alphabétique des radicaux des noms communs; la terminaison du ième radical nom commun est en ième position de ce fichier.

. FSEMNCOM: file of SEMNCDESC;

Fichier sémantique des noms communs, trié par ordre alphabétique des radicaux des noms communs correspondant

. BUFFNOM : SEMNCDESC;

Variable intermédiaire qui peut contenir un article du fichier FSEMNC.

. FCLASSE : file of VALCLAS;

Fichier des classes noms.
Le ième article de ce fichier contient les numéros des noms communs de la classe i. La classe 4 est vide. C'est la classe des noms propres. Je me suis aperçu trop tard qu'il s'agissait d'un cas particulier.

. LGCLASSE: array[1..22]of integer;

LGCLASSE i est le nombre d'éléments de la classe i.

e. Les variables spécifiques à la manipulation des verbes en E IS S

. TEVERBE : TABSENBV , TISVERBE : TABSISNBV , TSVERBE : TABSSNBV

T*VERBE[i] contient

- la terminaison infinitive
- le numéro d'ordre du groupe de conjugaison

du ième verbe en E, IS, S par ordre alphabétique.

. NUMEV, NUMISV, NUMSV : integer;

Numéro du plus grand verbe en E,IS,S.

. FESTXVB, FISSTXVB, FSSTXVB : file of SUFFVB;

Fichier des terminaisons des verbes triés respectivement par ordre alphabétique des radicaux des verbes.

. FSEMEVB, FSEMISVB, FSEMSVB : file of SEMVBDESC;

Fichiers sémantiques des verbes en E,IS,S triés chacun par ordre croissant des numéros d'ordre alphabétique des radicaux des verbes en E,IS,S.

. BUFFVERBE : SEMVBDESC;

Variable intermédiaire qui peut contenir un article d'un des trois fichiers ci-dessus.

. TERM : TERMCONJ;

Fichier des conjugaisons trié par ordre croissant des numéros de conjugaison du guide BESCHERELLE.

. TERMINAISON: TABTERM;

Tableau compacté des terminaisons, bout à bout sans séparation. Le premier caractère est '@'.

. INDIRECT : TABINDIR;

Tableau d'adresses des terminaisons dans TERMINAISON

f. Variables spécifiques à la manipulation des adjectifs qualific.

. TADJQ : TABSNBADJQ;

TADJQ[i] contient

- les différentes formes de terminaisons
- l'adresse dans INDICE du ième adjectif par ordre alphabétique.

. NUMADJ : integer;

Numéro du plus grand adjectif qualificatif

. FSTXADJ: file of SUFFIXE;

Fichier des terminaisons des adjectifs, trié par ordre alphabétique des radicaux des adjectifs.

. FSEMADJ: file of SEMADJDESC;

Fichier sémantique des adjectifs (il n'est pas implémenté). Le ième article de ce fichier donne la liste des classes adjectifs auxquelles appartient le ième adjectif par ordre alphabétique des radicaux d'adjectifs

g. Variables spécifiques à la manipulation des adverbes de manières

. TADV : TABINBADV;

TADV[i] est l'adresse dans INDICE du ième adverbe de manière par ordre alphabétique des adverbes.

. NUMADV: integer;

Numéro du plus grand adverbe de manière.

h. Variables spécifiques à la génération

NOVERBE : integer;

Numéro du verbe de la phrase.

NOVERBE = 1..NUMEV / 1..NUMISV / 1..NUMSV.

NG : integer;

Numéro du groupe EISS du verbe de la phrase

NG = 1 si c'est un verbe en E
2 ..... IS
3 ..... S.

TEMPS : integer;

Temps de l'indicatif du verbe de la phrase

TEMPS = 1 au présent
2 à l'imparfait — temps simples
3 au passé simple
4 au futur simple
5 au passé composé
6 au plus-que-parfait — temps composés
7 au passé antérieur
8 au futur antérieur.

PERS : integer;

Personne à laquelle est conjugué le verbe de la phrase

PERS = 1 à la 1ère singulier
2 .... 2ème .....
3 .... 3ème .....
4 .... 1ère pluriel
5 .... 2ème .....
6 .... 3ème .....

AUX : string;
Auxiliaire du verbe conjugué
Auxiliaire du verbe conjugué si le temps est composé,
indéfini si le temps est simple.

VERBCONJ : string;

Verbe conjugé si le temps est simple,
participe passé si le temps est composé.

DSUJET : SUJETDESC;

Sujet de la phrase.

FONCCPL : FONC;

Fonction du complément de la phrase,

COD, COIA, ADV.

COMPL : GRNOM;

Complément de la phrase.

NEXTLETTRE : char;

1ère lettre du mot suivant,nécessaire dans des situations comme par exemple :

LE ANIMAL qui doit devenir L'ANIMAL.

NBRCOI : char;

NBRCOI = 'S' si le coi est singulier
'P' si le coi est pluriel
indéfini s'il n'y a pas de coi dans la phrase.

PHRASETYP : integer;

PHRASETYP = i si la phrase est du ième type, i = 1..8,

cf diagrammes syntaxiques.

SPECIFICATIONS DES PROCEDURES DU PROGRAMME 'GENERATION'

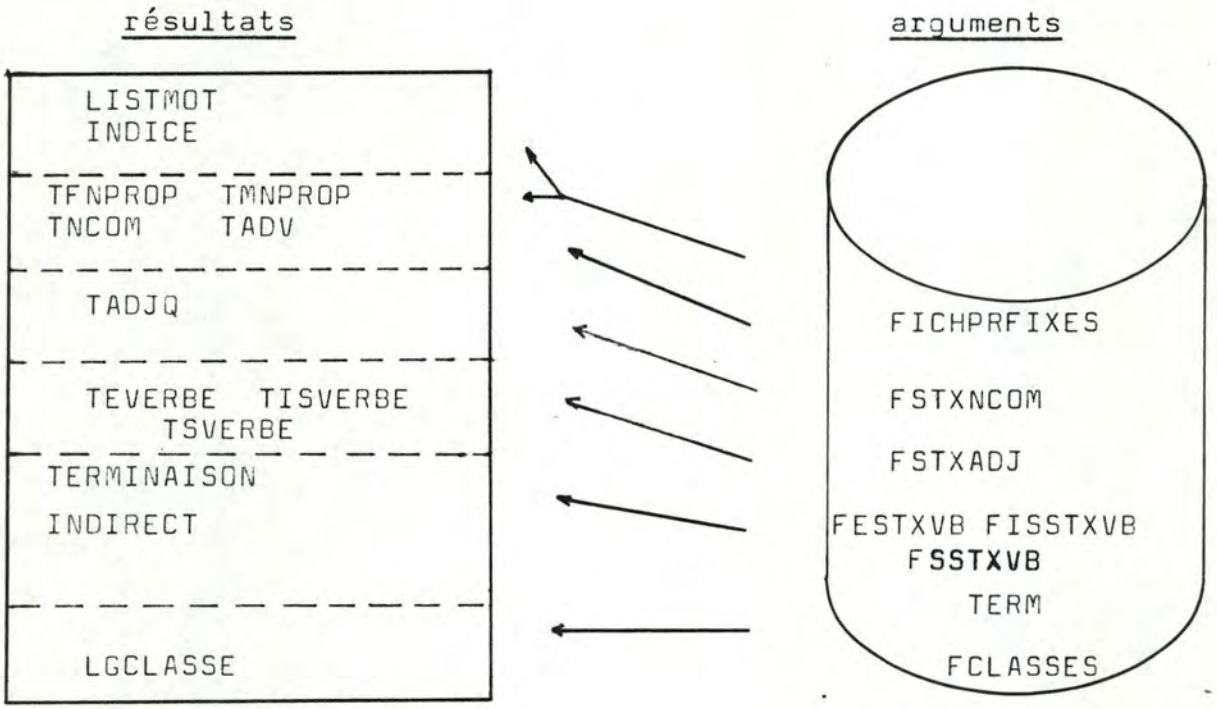
Le programme de génération se compose de 4 phases principales.

- Chargement en mémoire centrale des données à accès rapide.
- Déterminer les paramètres de la phrase à générer.
- Construire la phrase.
- Ecrire la phrase.

Voici successivement les procédures qui réalisent ces traitements.

a. Chargement des données.

Rappelez-vous que la plupart des informations à accès très fréquents (plus de 5 accès entre 2 sorties vers l'utilisateur) doivent être chargées en mémoire centrale pour ne pas trop diminuer le temps de réponse.



. SEGMENT PROCEDURE CHARGER;

(SEGMENT signifie que le code objet de cette procédure ne figure en mémoire centrale que durant l'exécution de la procédure. Je ne répéterai plus cette remarque.)

arguments : FICHPREFIXES, FSTXNCOM, FSTXADJ, FESTXVB, FISSTXVB, FSSTXVB, TERM, FCLASSES

résultats : LISTMOT, INDICE, TNCOM, TADJQ, TEVERBE, TISVERBE, TSVERBE, TFNPROP, TMNPROP, TADV, TERMINAISON, INDIRECT, LGCLASSE

fonction : Je l'ai dit,il s'agit de charger en mémoire centrale le contenu des fichiers arguments dans les tableaux résultats et d'y mettre à jour les pointeurs.

Cette procédure se compose de 3 procédures **indépendantes** :

- CHARGEMENT;
- CHARGTERM;
- CHARGLONGUEUR;

1. CHARGEMENT;

arguments : FICHPREFIXES, FSTXNCOM, FSTXADJ, FESTXVB, FISSTXVB, FSSTXVB

résultats : LISTMOT, NBRECAR
INDICE, NBREMOT
TNCOM, NUMNC
TADJQ, NUMADJ
TEVERBE, NUMEVB
TISVERBE, NUMISVB
TFNPROP, NUMFNP
TMNPROP, NUMMNP
TADV, NUMADV

fonction : Recopier les fichiers arguments dans les tableaux résultats, y mettre à jour les différents pointeurs.

idée de solution

Pour chaque article de FICHPREFIXES, p ex le ième faire

- charger FICHPREFIXES.ORTHO caractère par caractère dans LISTMOT à partir de la 1ère position libre (NBCHAR), le faire suivre du délimiteur ' '.

- garnir INDICE[i].ADRESSE avec NBCHAR;
mettre NBCHAR à jour (NBCHAR := NBCHAR + longueur de FICHPREFIXES.ORTHO + 1).

- garnir INDICE[i].NAT avec FICHPREFIXES.NAT.
si la nature du nouveau mot est

FNPROPRE : garnir TFNPROP
MNPROPRE : garnir TMNPROP
NCOM : garnir TNCOM
EVERBE : garnir TEVERBE
ISVERBE : garnir TISVERBE
SVERBE : garnir TSVERBE
ADJQ : garnir TADJQ
ADV : garnir TADV

```
garnir TFNPROP;

  NUMFNP:=NUMFNP + 1
  INDICE[i].NUMPARNAT:=NUMFNP
  TFNPROP [NUMFNP] :=i


garnir TMNPROP;

  NUMMNP:=NUMMNP + 1
  INDICE[i].NUMPARNAT:=NUMMNP
  TMNPROP[NUMMNP] :=i


garnir TNCOM;

  NUMNC:=NUMNC + 1
  INDICE[i].NUMPARNAT:=NUMNC
  TNCOM[NUMNC].ADIND :=i
  TNCOM[NUMNC].STXNCDESC := NUMNCième article de FSTXNCOM


garnir T*VERBE          * = E, IS, S

   NUM*V:= NUM*V + 1
   INDICE[i].NUMPARNAT:=NUM*V
   T*VERBE[NUM*V].ADIND := i
   T*VERBE[NUM*V].SUFFVB := NUM*Vième article de F*STXVB


garnir TADJQ

   NUMADJ := NUMADJ+1
   INDICE[i].NUMPARNAT:=NUMADJ
   TADJQ[NUMADJ]. ADIND :=i
   TADJQ[NUMADJ].SUFFIXE:= NUMADJième article de FSTXADJ


garnir TADV

   NUMADV := NUMADV+1
   INDICE[i] := NUMADV
   TADV[NUMADV] := i
```

2. CHARGTERM

arguments : TERM : file of TERMCONJ

résultats : TERMINAISON
INDIRECT

fonction : charger toutes les terminaisons par ordre successivement

- de numéro de groupe
- de temps présent
  imparfait
  passé simple

futur simple
participe passé

- de personne 1,2,3è singulier,1,2,3è pluriel

les unes derrières les autres dans TERMINAISON
garnir INDIRECT pour que:
INDIRECT i = adresse du début de la terminaison

idée de solution

tant que le fichier TERM n'est pas entièrement traité, faire

charger un groupe

charger un groupe =

charger	présent	= copier 6 conjugaisons dans TERMINAISON INDIRECT
	imparfait	.....
	passé simple	.....
	futur simple	.....
	particip pas.	..... 1 .....

3. CHARGLONGUEUR

arguments : FCLASSES

résultats : LGCLASSE

fonction : Recopier dans LGCLASSE les longueurs des classes du fichier FCLASSES pour que
LGCLASSE i = longueur de la classe en ième position du fichier FCLASSE

Le code du programme me semble assez simple à comprendre.

b. Déterminer les paramètres de la phrase à générer

Je tiens à rappeler d'abord quels sont les paramètres qui importent lors de la construction d'une phrase.

Ce sont le sujet, le verbe, le type de phrase.

le sujet: il importe de déterminer

- le genre
- le nombre
- la personne
- le type (pronom,suite,groupe nom)

le verbe: il importe de déterminer

- temps de l'indicatif
- groupe du verbe E, IS, S
- nombre de verbes
  (je ne suis plus très partisans de mettre plusieurs verbes à la file dans une phrase, c'est déjà assez compliqué comme ça)

le type de phrase: Il y en a 8 (cf diagrammes syntaxiques)

. SEGMENT PROCEDURE DETPHRASE;

arguments : -

résultats : DSUJET.NOMBRE 'M' ou 'P'
DSUJET.GENRE 'F' ou 'M'
DSUJET.PERSONNE '1' ou '2' ou '3'
DSUJET.VAL.TYPSUJ 'P' (pronom) ou 'G' (groupe nom)
ou 'S' (suite de composants)

NOVERBE
NG 1 ou 2 ou 3
TEMPS 1 .. 8

PHRASETYP 1 .. 8

fonction : Attribuer une valeur à chaque paramètre pour arriver à définir la phrase à générer

idée de solution : La lecture du programme me semble assez simple, j'ai d'ailleurs laissé dans le texte en commentaire les messages employés pendant les tests.

Une seule remarque: normalement, ce sont les scénarios qui guident le choix des paramètres. Comme ils ne sont pas implémentés, c'est le hasard qui guidera ce choix.

c. Construire la phrase déterminée

Pour chacun des 8 types de phrase il existe une procédure, GENPH1,GENPH2,GENPH3,GENPH4,GENPH5,GENPH6,GENPH7,GENPH8.

Toutes ces procédures **utilisent** les procédures d'un ensemble qui leur permet de

- construire un groupe nom
- accéder à un verbe
- conjuguer un verbe
- construire un pronom personnel
- accéder à un nom propre, à un adverbe

...

Voici ces procédures

. PROCEDURE CHARBESCH

arguments : SOURCE indice dans TERMINAISON de la première lettre de la terminaison à charger
LONGUEUR longueur du mot à charger

résultat : DESTINATION variable de type string où est chargée la terminaison

fonction : Charger dans DESTINATION la terminaison dont le 1er caractère se trouve dans TERMINAISON [SOURCE]

. PROCEDURE CHARBYTE

arguments : SOURCE indice dans LISTMOT de la première lettre du radical à charger
LONGUEUR longueur du radical à charger

résultat : DESTINATION variable de type string où est chargé le radical

fonction : Charger dans DESTINATION le radical dont le 1er caractère se trouve dans LISTMOT SOURCE

. PROCEDURE CHOISIRALEAT

arguments : BI, BS bornes inférieure et supérieure

résultat : RES

fonction : RES est uniformément distribué entre BI et BS incluses

. PROCEDURE PREMLETTRE

argument : MOT

résultat : LETTRE

fonction : LETTRE est la première lettre de MOT

. PROCEDURE CHOISIADV

arguments : LISTE liste de au plus 10 numéros d'adverbe rangés consécutivement dans un tableau d'entiers
LGLISTE longueur de LISTE

résultat : MOT

fonction : Choisir un numéro au hasard parmi la liste proposée et charger l'adverbe ayant ce numéro dans MOT

. PROCEDURE CHOISFNPROPRE

argument : NUMFNP

résultat : MOT

fonction : Choisir un numéro au hasard entre 1 et NUMFNP, et charger le nom propre féminin ayant ce numéro dans MOT

. PROCEDURE CHOISMNPROPRE

argument : NUMMNP

résultat : MOT

fonction : Choisir un numéro au hasard entre 1 et NUMMNP, et charger le nom propre masculin ayant ce numéro dans MOT

. PROCEDURE ACCEDE

arguments : NG numéro d'ordre d'un groupe Bescherelle
TP numéro de temps 1 présent
2 imparfait
3 passé simple
4 futur simple
5 participe passé
NP numéro de personne 1 .. 6

résultat : RES terminaison demandée

fonction : Chercher la terminaison d'un verbe de numéro de groupe NG, conjugué à la NPième personne du temps TP et mettre le résultat dans RES

idée de solution :

Nous savons qu'il y a 25 terminaisons par groupe , 4 temps de 6 terminaisons et un participe passé.

Si la conjugaison demandée est le participe passé, alors l'adresse du début de cette terminaison dans TERMINAISON est :

INDIRECT NG X 25

Si c'est un des 4 temps qui est demandé , alors l'adresse du début de la terminaison demandée dans TERMINAISON est :

INDIRECT NG-1 X 25 + TP-1 X 6 + NP

. PROCEDURE CONJVERBE

arguments : NOVERBE numéro du verbe
GPVERBE numéro de groupe EISS
TEMPS 1 présent
2 imparfait
3 passé simple
4 futur simple
5 passé composé
6 plus-que-parfait
7 passé antérieur
8 futur antérieur
PERS numéro de personne 1 .. 6

résultats : AUX si le temps est simple : indéfini
composé : auxiliaire du vert
VERBCONJ si le temps est simple : verbe conjugué
composé : participe passé
NEXTLETTRE première lettre de VERBCONJ si le temps est simple, ou de AUX si le temps est composé

fonction : conjuguer le verbe de numéro NOVERBE ,de groupe GPVERBE, à la personne PERS du temps TEMPS;
Renvoyer la première lettre de AUX (temps composé) ou de VERBCONJ (temps simple)

idée de solution :

Selon le groupe EISS du verbe, chercher dans le tableau de verbes correspondant le numéro d'ordre de son groupe de conjugaison et accéder au radical.

Connaissant le numéro d'ordre de son groupe de conjugaison, le temps et la personne :

si le temps est simple

Accéder directement à la terminaison et la concaténer au radical

si le temps est composé

Accéder à la terminaison de l'auxiliaire pour ces paramètres (le verbe avoir de groupe 1) , garnir AUX

Accéder à la terminaison participe passé et la concaténer au radical, garnir VERBCONJ.

Ne pas oublier PREMLETTRE.

. PROCEDURE CONSTRPROPERS

arguments : SJ.NOMBRE 'S' ou 'P'
SJ.PERSONNE '1' ou '2' ou '3'
SJ.GENRE 'F' ou 'M' ou 'I' (I= indéfini ='on')
NEXTLETTRE 1ère lettre du mot qui suivra le pronom

résultat : SJ.VAL.PRONOM

fonction : SJ.VAL.PRONOM sera un pronom personnel ou un prénom qui répond aux exigences passées en arguments.

En lisant le programme, vous constaterez que les pronoms ne sont pas issus d'une base de données mais affectés directement. Cela me semble plus efficace, mais je n'ai pas pu tester cette méthode pour le décodage de phrase; je n'ai envisagé que la génération

. SEGMENT PROCEDURE CONSGRPNOM

C'est la procédure la plus longue du programme génération, c'est aussi la plus difficile.

Elle construit un groupe nom.

arguments : NOMBRE 'S' si le groupe nom est singulier
'P' ..... pluriel
GENRE 'F' ..... féminin
'M' ..... masculin
TYPEVB 1 si le groupe nom accompagne un verbe en E
2 ..... IS
3 ..... S
INDVERBE numéro d'ordre alphabétique du verbe parmi les verbes de son groupe
FONCTION fonction que jouera ce groupe nom auprès du verbe
FONCTION = SUJ
COD
COIA

résultat : GRN l'enregistrement du groupe nom

il comporte 6 champs

Les 5 premiers champs contiennent dans l'ordre de leur écriture les mots du groupe nom

GRN.DETERMINANT.D1
GRN.DETERMINANT.D2
GRN.ADAV
GRN.NOM
GRN.ADAP

Le 6ième champs est un indicateur à 5 positions

GRN.INDIC [1] = 0 si le champ est vide
1 ..... un adject.défini
2 ..... un adject.indéfini
3 ..... un adject.possesif
4 ..... un adject.démonst.

GRN.INDIC [2] = 0 si le champ est vide
1 ..... un adject.numéral

GRN.INDIC [3] = 0 si le champ est vide
1 ..... un adject.qualific.

GRN.INDIC [4] = il contient un nom commun

GRN.INDIC [5] = 0 si le champ est vide
1 ..... un adject.qualific.

attention ! il y a des contraintes

GRN.INDIC [1] = 0 ⟹X GRN.INDIC [2] = 1
GRN.INDIC [1] = 2 ⟹ GRN.INDIC [2] = 0
GRN.INDIC [2] = 1 ⟹ GRN.INDIC [1] ≠ 2
GRN.INDIC [3] = 1 ⟹ GRN.INDIC [5] = 0
GRN.INDIC [5] = 1 ⟹ GRN.INDIC [3] = 0
GRN.INDIC [2] = 0 ⟹X GRN.INDIC [1] ≠ 0

FONCTION

Dans certains cas, FONCTION est aussi un résultat. Parce que pour la fonction demandée, je ne peux trouver très vite un nom commun ayant toutes les caractéristiques exigées, je choisi de changer la fonction. Ceci n'est vrai que pour les fonctions COD et COIA qui ne sont que des garnitures pour la phrase.

Pourquoi je ne trouverais pas très vite un nom commun?

Il ya deux raisons

- j'ai omis de rentrer un tel mot dans la base
- l'étude de la solution vous apprendra que je n'étudie pas tous les mots possibles pour gagner du temps

```
traitement des erreurs

si FONCTION = COD

   alors si le verbe n'admet pas de COD et admet un COI
           alors FONCTION = COIA

           sinon
           (le verbe n'admet ni COD ni COIA)
                FONCTION = ADV
                INDIC[1],INDIC[2],INDIC[3],INDIC[5] = 0
                INDIC[4] = 1


si FONCTION = COIA

   alors si le verbe n'admet pas de COIA et admet un COI
           alors FONCTION = COD

           sinon
           (le verbe n'admet ni COD ni COIA)
                FONCTION = ADV
                INDIC[1],INDIC[2],INDIC[3],INDIC[5] = 0
                INDIC[4] = 1
```

idée de solution et structure du programme :

niveau-1

a. déterminer le chemin du groupe nom

   . de quoi se composera-t-il ?

     adjectif qualificatif avant le nom, après, nom seul ?
     si nombre sing, le déterminant sera-t-il un adjectif
     défini, indéfini, possesif,démonstratif??
     si nombre pluriel, le déterminant sera-t-il un adjectif
     défini, indéfini, possessif, démonstratif, numéral ?
     si c'est un numéral, sera-t-il seul ou suivra-t-il un
     adjectif défini, possessif ou démonstratif ?

   . enregistrer ce chemin dans un indicateur

b. rassembler les caractéristiques syntaxiques que devra posséder le nom commun du futur groupe

   . nombre
   . genre
   . adjectif avant, après, pas d'adjectif (qualificatif)

c. construire un groupe nom autour de ces caractéristiques, en répondant si possible à la fonction demandée pour le verbe exigé

niveau-2

a. construire un groupe nom autour de ces caractéristiques

. extraire de la base de données un mot ayant les caractéristiques demandées et l'accorder

. entourer ce mot des attributs déterminés au niveau-1 et les accorder

niveau-3

a. extraire de la base de données un mot ayant les caractéristiques demandées et l'accorder

. accéder à la liste des numéros de classes dans lesquelles je peux choisir un mot

. accès au fichier sémantique des verbes de type TYPEVB et précisément au verbe de ce fichier numéro INDVERBE

. retenir la liste correspondante à la fonction FONCTION si cette liste est vide, voir le traitement d'erreurs à la page précédente

. parmi cette liste, rechercher au travers des différentes classes le mot convoité

niveau-4

a. parmi cette liste, rechercher au travers des différentes classes le mot convoité

.Pour une raison de performances, il n'est pas possible de comparer les caractéristiques de chaque mot de chaque classe permise aux caractéristiques demandées.

Je vais donc retenir dans la liste pas plus de trois classes dans lesquelles j'étudierai tous les mots. Pourquoi trois ? Parce que dans ma base de données, c'est le nombre qui allie le mieux rapidité, risque de ne pas trouver et équiprobabilité de tirage des mots (les classes sont de ≠ longueurs).

Ces trois classes sont tirées au hasard, mais au prorata du nombre de mots qu'elles contiennent.

.Parcourir les classes sélectionnées, chaque fois qu'un mot est valable je note sa référence dans un tableau.
Il me reste, quand j'ai parcouru tous les mots, à choisir une référence au hasard parmi les références du tableau.

.Accéder au radical de ce mot et l'accorder

. Spécifications des procédures internes à CONSGRPNOM

. PROCEDURE DETCHEMGRPNOM

arguments : NOMBRE 's' ou 'P'
GENRE 'F' ou 'M'

résultat : CHEMIN

CHEMIN[1]= 0 si il n'y a ni adj.défini,indéf,poss,dém
1 si il y a un adj.défini
2 .....indéfini
3 .....possesif
4 .....démonstratif

CHEMIN[2]= 0 si il n'y a pas d'adjectif numéral
1 si il y a un adjectif numéral

CHEMIN[3]= 0 si il n'y a pas d'adj.qual. avant le nom
1 si il y a un adj.qual avant le nom

CHEMIN[4]= 1 il y a toujours un nom commun

CHEMIN[5]= 0 si il n'y a pas d'adj.qual. après le nom

contraintes

CHEMIN[1] = 0 ⇒ CHEMIN[2]= 1
CHEMIN[1] = 2 ⇒ CHEMIN[2]= 0
CHEMIN[2] = 0 ⇒ CHEMIN[1]≠ 0
CHEMIN[3] = 1 ⇒ CHEMIN[5]= 0
CHEMIN[5] = 1 ⇒ CHEMIN[3]= 0
CHEMIN[2] = 1 ⇒ CHEMIN[1]≠ 2

fonction : garnir CHEMIN avec les valeurs appropriées selon les valeurs des arguments

idée de solution : Comme je laisse dans le texte du programme les messages entrée sortie employés durant les tests, le programme me semble suffisamment lisible que pour encore le commenter.

. PROCEDURE GARNIR

arguments : NOMBRE, GENRE, CHEMIN

résultat : CARACT tableau de booleens

CARACT[1] = vrai si NOMBRE='S' et GENRE='F',faux sinon
[2] = ...'P' .....'F',.....
[3] = ...'S' .....'M',.....
[4] = ...'P' .....'M',.....
[5] = ... CHEMIN[3]= 1 ,faux sinon
[6] = ... CHEMIN[5]= 1 ,.....

fonction : rassembler les différentes caractéristiques du mot convoité pour faciliter les comparaisons futures

. PROCEDURE CONSTRUIREGRPNOM

Le coeur du problème, tous les paramètres ont été déterminés. Il va falloir trouver un nom commun adéquat, l'accorder et l'entourer de ses déterminants,adjectifs autres.

arguments :NOMBRE, GENRE, FONCTION, CARACT, TYPEVB, INDVERBE

les spécifications de ces variables sont les mêmes que celles présentées dans CONSGRPNOM et GARNIR.

résultats :GRN , FONCTION

même remarque que pour les arguments

Pour construire le groupe nom demandé, **CONSTRUIREGRPNOM** va employer quelques procédures internes.

Les voici ...

. PROCEDURE CHERCHENOM

arguments : FONCTION, CARACT, TYPEVB, INDVERBE, NOMBRE, GENRE

même remarque que pour les arguments de la **procédure** précédente

résultats : **NOM nom convoité**
IDXCHOIX numéro du nom trouvé
FONCTION fonction que joue ce nom près du verbe

fonction : Charger dans NOM un nom commun répondant aux exigences de syntaxe (CARACT) et de sémantique (FOCTION,TYPEVB, INDVERBE) passées en arguments.
Charger dans IDXCHOIX le numéro de ce nom commun.
Si la fonction a changé en cours de route, mettre FONCTION à jour.

idée de solution :

- Ouvrir le fichier sémantique des verbes de type TYPEVB
- Accéder au INDVERBEième article
- Extraire la liste de classes correspondant à la fonction FONCTION
- Choisir trois classes parmi cette liste
- De ces trois classes, extraire un mot de caractéristique CARACT
- Accorder ce mot, le charger dans NOM, mettre son numéro dans IDXCHOIX
- si les tirets 4 ou 5 n'étaient pas faisables, changer la fonctior FONCTION.

. PROCEDURE CHOISIRMOT

arguments : LISTCLASSE liste de classes de mots
NBCLASSE longueur de LISTCLASSE
CARACT caractéristique du mot recherché

résultat : IDXCHOIX numéro d'un nom commun

fonction : Sélectionner un mot dans une liste de listes de mots LISTCLASSE. Le mot doit avoir les caractéristiques CARACT. Son numéro sera noté dans IDXCHOIX
Si aucun mot ne correspond à CARACT, IDXCHOIX = 0.

idée de solution :

- Tirer au plus trois classes au hasard

  Il faut que les mots des différentes classes aient tous autant de chances de sortir. Un simple tirage au sort de trois classes ne convient pas, il avantagerait les mots des petites classes.

  Je vais compter le nombre total de mots parmi toutes les classes Je choisirai trois fois de suite une valeur entre 1 et ce total. Après chaque tirage, je retiendrai la classe qui possède le valeurième mot en comptant à partir de la première de la liste et en passant de l'une à l'autre. Cette classe retenue sera ôtée de la compétition pour les prochains tirages, sa longueur sera déduite du total.

- Parcourir la description de chacun des mots de ces classes. Quand un mot convient, charger son numéro dans un grand tableau.

- Tirer un numéro au hasard dans ce tableau et en garnir IDXCHOIX. Si le tableau est vide, IDXCHOIX = 0

. PROCEDURE TOTALMOT

arguments : LISTCLASSE, NBCLASSE cf arguments de CHOISIRMOT

résultat : TOTAL nombre de mots parmi les NBCLASSE de LISTCLASSE

fonction : compter les mots de LISTCLASSE

. PROCEDURE CHOISICLASSE

arguments : LISTE liste de classes de mots
TOTAL nombre de mots parmi toutes les classes

résultat : NOCL indice dans LISTE

fonction : trover NOCL tel que LISTE[NOCL] est la classe choisie

. PROCEDURE COMPACTE

arguments : LISTE liste de classes
LGLISTE longueur de LISTE
NUMERO indice dans LISTE
résultats : LISTE liste de classes
LGLISTE longueur de LISTE

fonction : LISTE i = ancienne LISTE i+1
pour i = NUMERO .. LGLISTE-1

. PROCEDURE REMPLTAB

arguments : NUM numéro de classe
CARACT caractéristique d'un nom commun

résultats : TAB50 tableau de numéros de mots ayant les caractéristiques CARACT
LGTAB longueur de TAB50

fonction : Charger les mots de la classe numéro NUM qui ont les caractéristiques CARACT dans le tableau TAB50, LGTAB est le nombre de mots chargés ou la longueur de TAB50.

. PROCEDURE COMPARE

arguments : CARDEM caractéristiques demandées pour le mot convoit
CAROBS caractéristiques du mot proposé

résultat : VALABLE vrai si CARDEM = CAROBS
faux sinon

fonction : dire si CARDEM égale CAROBS

idée de solution : Nous avons 2 tableaux de mêmes types. Parcourons les ensembles, si 2 éléments de même indice different, on stoppe ils ne sont pas égaux.

. PROCEDURE ACCORDMOT

arguments : INDMOTCHOISI numéro du nom commun trouvé
NOMBRE nombre du nom, 'S' ou 'P'
GENRE genre du nom, 'F' ou 'M'

résultat : MOT le mot choisi accordé

fonction : Accorder en genre et en nombre le nom commun de numéro INDMOTCHOISI, charger le résultat dans MOT.

idée de solution : Accéder au radical du nom commun, accéder à la terminaison en genre et nombre correspondant et concaténer les deux.

. PROCEDURE TRTERREUR

argument : FONCTION fonction que le mot doit exercer près du verb

résultats : FONCTION fonction de remplacement
LISTE liste de classes de remplacement
LGLISTE longueur de LISTE
CHEMIN composants du groupe nom

fonction : Si la fonction était COD

alors si le verbe admet des COIA

alors FONCTION = COIA

sinon FONCTION = ADV
modifier CHEMIN

sinon (la fonction était COIA)

si le verbe admet des COD

alors FONCTION = COD

sinon FONCTION = ADV
modifier CHEMIN

Modifier CHEMIN est un traitement d'exception, il est à la limite de l'admissible, je l'emploie parce que je considère l'adverbe comme un complément.

Traiter l'adverbe comme un groupe nom d'un seul mot est une facilité qui me permet de ne pas m'occuper plus tard de ce changement lors de l'écriture de la phrase.

La réentrance de la variable FONCTION est le seul point noir de ce programme, elle reste pour moi la seule méthode rapide de correction de ce type d'erreur Dans une application où le temps de réponse est très important, j' ai bien dû l'appliquer.

La fin justifie les moyens !

. PROCEDURE ADDEFGEN *

arguments : NOMBRE, GENRE
MOTQUISUIT mot qui suivra l'adjectif défini

résultat : RESULTAT adjectif défini du nombre et du genre passés en arguments

fonction : mettre dans RESULTAT un adjectif défini de genre et de nombre précisés en arguments

* J'estime que les cinq procédures qui vont suivre sont assez simples à comprendre et ne méritent pas d' explication de l'idée de solution

. PROCEDURE ADINFGEN

arguments : NOMBRE, GENRE

résultat : RESULTAT adjectif indéfini de nombre et de genre passés en arguments

. PROCEDURE ADPOSGEN

arguments : NOMBRE, GENRE, MOTQUISUIT

résultat : RESULTAT adjectif possessif de nombre et de genre passés en arguments

. PROCEDURE ADDEMGEN

arguments :NOMBRE, GENRE, MOTQUISUIT

résultat :RESULTAT adjectif démonstratif de nombre et de genre passés en arguments

. PROCEDURE ADNUMGEN

arguments : -

résultat : RESULTAT égale 'UN','DEUX','TROIS','QUATRE','CINQ' 'SIX','SEPT','HUIT','NEUF','DIX' au hasard.

. PROCEDURE ACCORDQUAL

arguments : ADRINDICE numéro d'adjectif qualificatif
NOMBRE, GENRE

résultat : ADJ adjectif qualificatif de numéro ADRINDICE accordé en nombre et en genre

fonction : renvoyer dans ADJ l' adjectif qualificatif de numéro ADRINDICE accordé selon NOMBRE et GENRE

idée de solution : Accéder au radical de l'adjectif, chercher la terminaison voulue dans TADJQ, enfin concatener le radical et la terminaison

. PROCEDURE GENAVQUAL

arguments : NUMMOT numéro du mot à faire précéder d'un adjectif qualificatif
NOMBRE, GENRE

résultat : ADJ adjectif qualificatif accordé en genre et en nombre

fonction : Extraire un adjectif qualificatif apte à précéder le nom commun numéro NUMMOT, l'accorder selon NOMBRE et GENRE et le charger dans ADJ.

idée de solution : trouver dans le NUMMOTième article du fichier sémantique des noms communs un numéro d'adjec-

tif qui puisse précéder le nom commun.
Employer la procédure ACCORDQUAL pour chercher son radical et l'accorder.

. PROCEDURE GENAPQUAL

arguments : NUMMOT, NOMBRE, GENRE cf GENAVQUAL

résultat : ADJ cf GENAVQUAL

fonction : Extraire un adjectif qualificatif apte à suivre le nom commun numéro NUMMOT, l'accorder selon NOMBRE et GENRE et le charger dans ADJ.

idée de solution : cf GENAVQUAL, une seule différence on cherche un adjectif qui puisse suivre le nom commun.

.... fin des procédures internes de CONSTRUIREGRPNOM ....

..... fin des spécifications de la procédure CONSGRPNOM .....

. SEGMENT PROCEDURE CONSSUITE

arguments : PERSONNE personne du sujet que forme la suite
GENRE genre du sujet qu'est la suite
TYPEVB type de verbe dont la suite est sujet
INDVERBE numéro du verbe de type TYPEVB

résultat : SUITE cf le type SUITCOMPOSANT dans les spécific. des types de données

fonction : Construire une suite de composants, 2 ou 3 au choix. Elle aura le genre et la personne passés en arguments et sera compatible sémantiquement avec le verbe numéro INDVERBE du groupe TYPEVB.

idée de solution :

- déterminer les paramètres de la suite
  - . nombre de composants
  - . personne,genre et type des composants
- construire la suite selon ces paramètres

1. déterminer les paramètres de la suite, il faut respecter des contraintes

   a. personne

La personne la plus petite l'emporte.
Si la personne est i ,la personne de chaque composant est ≥ i

ex i = 2 (2ème pers) alors aucun composant ne peut être à la 1ère personne

b. type de composant

Un composant peut être pronom ou groupe nom. Seul un composar à la 3ème personne peut être de type groupe nom.

c. genre

Le masculin l'emporte sur le féminin.
Si le genre est féminin, tous les composants doivent être féminin.
Si le genre est masculin, il faut au moins un masculin parmi les composants

Connaissant ces 3 contraintes, la procédure DETSUITE dans laquelle j'ai laissé les messages employés lors des tests me semble assz simple à comprendre.

. PROCEDURE DETSUITE

arguments : PERSONNE, GENRE

résultat : SUITE.COMPi.PC personne du ième composant
SUITE.COMPi.NC nombre .....
SUITE.COMPi.GC genre .....
SUITE.COMPi.TC type .....
SUITE.NBCOMP nombre de composants 2 ou 3

i = 1 .. SUITE.NBCOMP

2. construire la suite selon ces paramètres

. PROCEDURE CONSTRSUITE

arguments : TYPEVB type du verbe dont la suite est sujet
INDVERBE numéro du verbe de type TYPEVB
SUITE seulement sa partie description
résultat : SUITE suite de composants en règle avec tous les paramètres

. PROCEDURE CONSPRONSUITE

arguments : NOMBRE, PERSONNE, GENRE

résultat : PRON pronom personnel d'une suite de composants

. PROCEDURE COMPGEN

arguments : COMPOSANT.TC, COMPOSANT.NC, COMPOSANT.GC,
COMPOSANT.PC,

résultat : COMPOSANT.CP si le composant est un pronom
COMPOSANT.CG si le composant est un groupe nom

... fin des spécifications de CONSSUITE ...

. PROCEDURE GENSUJ

arguments : DSUJET donnée de type SUJETDESC cf spécif.types
.VAL.TYPSUJ
.PERSONNE
.GENRE
.NOMBRE
TYPEVB, INDVERBE

résultat : DSUJET.VAL.PRONOM
DSUJET.VAL.GR
DSUJET.VAL.ST

fonction : attribuer une valeur à résultat selon le type de sujet demandé et les arguments d'accord

. PROCEDURE GENCODCOIA

arguments : INDVERBE numéro du verbe à compléter
TYPEVB type du verbe

résultats : NOMBRE le nombre du complément
FONCTION la fonction du complément
COMPL le complément

fonction : Construire un complément au choix COIA ou COD, l'enregistrer dans COMPL.
Renvoyer sa fonction et son nombre

d. Ecrire la phrase que l'on vient de construire

- Deux procédures de mise en page

. PROCEDURE PASSAGEALALIGNE

arguments : LONGLIGNE longueur courante de la ligne
LONGAJOUTEE longueur du prochain mot à écrire

résultat : On passe à la ligne si il n'y a plus assez de place pour écrire le prochain mot

fonction : La largeur utile de l'écran APPLE II est de 40 caractères.
Cette procédure envoie un carriage return chaque qu'elle calcule que LONGLIGNE+LONGAJOUTEE 40.
LONGLIGNE représente toujours la longueur courante de la ligne .

. PROCEDURE BLANC

arguments : LONGLIGNE longueur courante de la ligne

résultat : écrire un blanc si on n'est pas en début de ligne
LONGLIGNE = LONGLIGNE + 1 si on n'est pas en début de ligne

- procédures d'écriture de la phrase

. PROCEDURE ECRPRONOM

argument : PRONOM variable de type string

résultat : PRONOM est écrit à l'écran

. PROCEDURE ECRGRNOM

argument : GRP variable de type grnom

résultat : les composants du groupe nom sont écrits

idée de solution :Parcourir l'indicateur GRP.INDIC du groupe nom chaque fois qu'un champ est renseigné comme pas vide, l'imprimer. Dans tous les cas les composants sont à imprimer dans l'ordre ou ils apparaissent.

. PROCEDURE ECRSUITE

argument : SUITE

résultat : SUITE est écrit

fonction : Ecrire les uns derrière les autres les SUITE.NBCOMP composants.

Si il y a deux composants

S'ils sont séparément chacun au singulier, alors ils seront séparés par 'ET', pour que la suite soi au pluriel sans ambiguité.

Si l'un des deux est au pluriel, ils seront séparé par 'ET' ou par 'OU' au choix.

Si il y a trois composants

Les deux premiers seront séparés d'une virgule. Si les trois composants sont au singulier, les deux derniers composants seront séparés par 'ET', sinon, ils seront séparés par 'ET' ou 'OU' au choi

. PROCEDURE ECRIRESUJET

argument : DSUJET.VAL.TYPSUJ type du sujet
DSUJET.VAL.PRONOM pronom
DSUJET.VAL.GR groupe nom
DSUJET.VAL.ST suite de composants

résultat : DSUJET.VAL.PRONOM ou DSUJET.VAL.GR ou DSUJET.VAL.ST est écrit

fonction : écrire le sujet décrit par DSUJET

. PROCEDURE ECRIREVERBE

argument : TEMPS numéro d'un temps de l'indicatif 1 .. 8
VERBCONJ verbe conjugué ou participe passé
AUX auxiliaire d'un verbe conjugué si le temps est composé

résultat : AUX est écrit si le temps est composé
VERBCONJ est toujours écrit.

fonction : Ecrire le verbe

si TEMPS $\leq$ 4 écrire VERBCONJ

si TEMPS $>$ 4 écrire AUX, un blanc, VERBCONJ

. PROCEDURE ECRIRECOIA

arguments : COIA groupe nom
NOMBRE nombre de COIA = 'S' ou 'P'

résultat : Le groupe nom est écrit

fonction : Ecrire le groupe nom en tenant compte de la prépositic

Il faut savoir que :

- A LE devient AU, A LES devient AUX
- A suivi d'un adjectif indéfini,possessif,démonstratif ne change pas
- devant un adjectif numéral on aura indifférement A ou AUX

. PROCEDURE ECRCOMPL

arguments : COMPL groupe nom
FONCTION fonction du complément COD,COIA, ADV
NOMBRE nombre du complément

résultat : le complément est écrit

idée de solution : employer la procédure ECRGRNOM pour les fonctions COD et ADV

employer la procédure ECRIRECOIA pour les fonctions COIA

... fin des spécifications du programme génération ...

ANNEXE A.III

SPECIFICATIONS DU SYSTEME DE CORRECTION AUTOMATIQUE DE FAUTES D'ORTHOGRAPHE D'USAGE ET DE FAUTES DE FRAPPE

- TYPES DE DONNEES
- VARIABLES
- PROCEDURES

SPECIFICATIONS DU SYSTEME DE CORRECTION AUTOMATIQUE DE FAUTES D'ORTHOGRAPHE D'USAGE ET DE FAUTES DE FRAPPE.

a. Spécifications des types de données et des constantes
b. Spécifications des variables globales employées
c. Idée de solution, architecture du programme
d. Spécifications des procédures

a. Spécifications des types de données et des constantes

. Les constantes

NBRMOT : longueur maximum du type de tableau TABIND ;
NBRECAR : longueur maximum du type de tableau TABCAR ;
NBRETERM : longueur maximum du type de tableau TABTERM .

. Les types

MATRICE

```
MATRICE=ARRAY ['@'..'Z','@'..'Z'] OF INTEGER;
```

Ce type de donnée représente un tableau de 27 lignes et 27 colonnes Il contient des entiers. Les indices de ce type de tableau sont des caractères ordonnés selon l'ordre ASCII, le '@' représente le séparateur qui précède et suit chaque mot. Notons que '@' précède le 'A' dans l'ordre ASCII.

TABCAR

```
TABCAR= PACKED ARRAY[1..NBRECAR] OF CHAR;
```

Ce type de tableau enregistre bout à bout tous les mots

- triés par ordre alphabétique
- le caractère '@' commence et fini ce type de tableau, et sépare chaque mot du suivant
- le dernier mot est bidon, c'est 'ZZ'

TABIND

TABIND= ARRAY[1..NBREMOT] OF INTEGER;

Tableau d'adresses. Le ième élément de ce type de tableau fournit l'adresse dans un tableau de type TABCAR du premier caractère du ième mot du vocabulaire, par ordre alphabétique.

TABTERM

TABTERM= ARRAY[1..NBRETERM]OF INTEGER;

Tableau d'adresses. Le ième élément de ce type de tableau fournit l'adresse dans un tableau de type TABCAR du 3ème caractère avant la fin du ième mot de plus de six lettres, selon l'ordre alphabétique.

Il reste quelques types dans le programme que je ne définis pas ici, vous découvrirez plus facilement leur utilité en connaissant les variables qui les emploient et le problème précis à résoudre.

b. Spécifications des variables globales du programme

DIGRAM : MATRICE;

Cette variable contient les valeurs des fréquences des digrammes du vocabulaire. Ce que c'est qu'un digramme? Vous avez mal lu le mémoire...

LISTMOT : TABCAR;

Tableau de caractères. C'est lui qui contient bout à bout, séparés d'un caractère spécial '@', tous les mots du vocabulaire classés par ordre alphabétique. Le premier caractère est '@' ainsi que le dernier. Le dernier radical est bidon, c'est 'ZZ'. Pourquoi ? Voyez à la page A.III.9 où je l'emploie.

INDICE : TABIND;

C'est un tableau d'indirection.
INDICE[i] fournit l'adresse dans LISTMOT du premier caractère du ième mot par ordre alphabétique des mots.

INDICETERM : TABTERM;

C'est un tableau d'indirection, INDICETERM[i] fournit

l'adresse dans LISTMOT du 3ième caractère avant la fin du mot de plus de 6 lettres dont la terminaison arrive en ième position par ordre alphabétique des terminaisons.

DIMCAR : integer;

Nombre de caractères de LISTMOT, soit la somme des longueurs de tous les mots, y compris 'ZZ', plus tous les séparateurs.

DIMMOT : integer;

Nombre de mots du dictionnaire, y compris 'ZZ'.

DIMTERM : integer;

Nombre de mots de plus de 6 lettres.

ENTREE : string 20 ;

Mot dont on souhaite éventuellement corriger l'orthographe.

ORTHO : boolean;

Variable booléene vraie si ENTREE correspond à un mot du dictionnaire, fausse sinon.

BI,BS : integer;

Adresses dans INDICE, ces deux variables sont des résidus d'une recherche dichotomique de ENTREE. Si la recherche a échoué BI et BS représentent deux adresses consécutives entre lesquelles devrait se trouver ENTREE si elle existait. En cas de succès de la recherche, le calcul (BI+BS) div 2 donne l'adresse de ENTREE dans INDICE.

COMMAX : integer;

Cette variable représente le plus grand nombre de premières lettres que ENTREE puisse avoir en commun avec un mot du dictionnaire.

POSDEPART : integer;

Index dans INDICE du mot qui possède le plus grand nombre de premières lettres communes avec ENTREE.
Il peut y avoir plusieurs mots ayant cette propriété. Ils seraient alors tous consécutifs. Du quel s'agit-il ? Cela n'a aucune importance.

FREQUENCE : TAB7;

TAB7 = array [1..7] of integer;

Pourquoi 7 ? Parce que nous n'employons cette variable que pour corriger des mots de longueur 6, et ayant donc au plus 7 digrammes.

FREQUENCE[i] = fréquence du ième digramme de ENTREE.

si i=1 La valeur du ième digramme est

DIGRAM ['@',ENTREE[i]] .

si i=2..COMMAX La valeur du ième digramme est

DIGRAM [ENTREE[i-1], ENTREE[i]]

si i=COMMAX+1 La valeur du ième digramme est

. Si COMMAX = longueur de ENTREE

DIGRAM [ENTREE COMMAX ,'@']

. Si COMMAX < longueur de ENTREE

DIGRAM [ENTREE[COMMAX],ENTREE [COMMAX+1]]

DIMFREQ : integer;

Indice du plus grand élément défini de FREQUENCE.

PROD : TAB6;

TAB6 = array[1..6]of integer;

Pourquoi 6 ? Parce que nous n'employons cette variable que pour corriger des mots de longueur égale à 6.

Tableau des produits des fréquence des digrammes.

PROD[i] = FREQUENCE[i] X FREQUENCE[i+1]

Si FREQUENCE i = 0

. Si i=1

PROD[1] = 0 et PROD[j] est indéfini si j>i

. Si i=2..COMMAX

PROD[i-1] = 0 et PROD[j] est indéfini si j>i
PROD[i] = 0

. Si i=COMMAX+1

PROD[COMMAX] = 0 et PROD[j] est indéfini si j≥i

DIMPROD : integer;

Indice du plus grand élément défini de PROD.

FREQNUL : integer;

Indice du 1er élément de FREQUENCE qui est nul, zéro si aucun élément de FREQUENCE n'est nul.

A noter :

Pour FREQUENCE

si FREQNUL = i alors FREQUENCE[j] ≠ 0 si j<i
et FREQUENCE[j] est indéfini si j>i

Pour PROD

si FREQNUL = 1 alors PROD[1] = 0
PROD[i] est indéfini si i>1

si FREQNUL = i i=2..COMMAX
alors PROD[i-1] = 0 et PROD[i] = 0
PROD[j] est indéfini si j>i

si FREQNUL = COMMAX+1
alors PROD[COMMAX] = 0
PROD[j] est indéfini si j>COMMAX

POSLETCRIT : TAB6;

Imaginez le tableau PROD indicé 1..DIMPROD, imaginez qu'on le trie par ordre croissant mais qu'au lieu de recopier les valeurs du tableau, on recopie les indices correspondants, vous obtiendriez un tableau d'indices: c'est POSLETCRIT.

POSLETCRIT[i] est le ième plus grand élément de PROD.
POSLETCRIT[i] est indéfini si i DIMPROD.

Ce tableau nous indique dans quel ORDRE il nous faut traiter les caractères de ENTREE pour 'chipoter' d'abord ceux qui dans nos statistiques sont présentés comme les moins souvent entourés de leurs deux voisins dans ENTREE.

DIMPOSLETCRIT : integer;

Plus grand élément défini de POSLETCRIT.

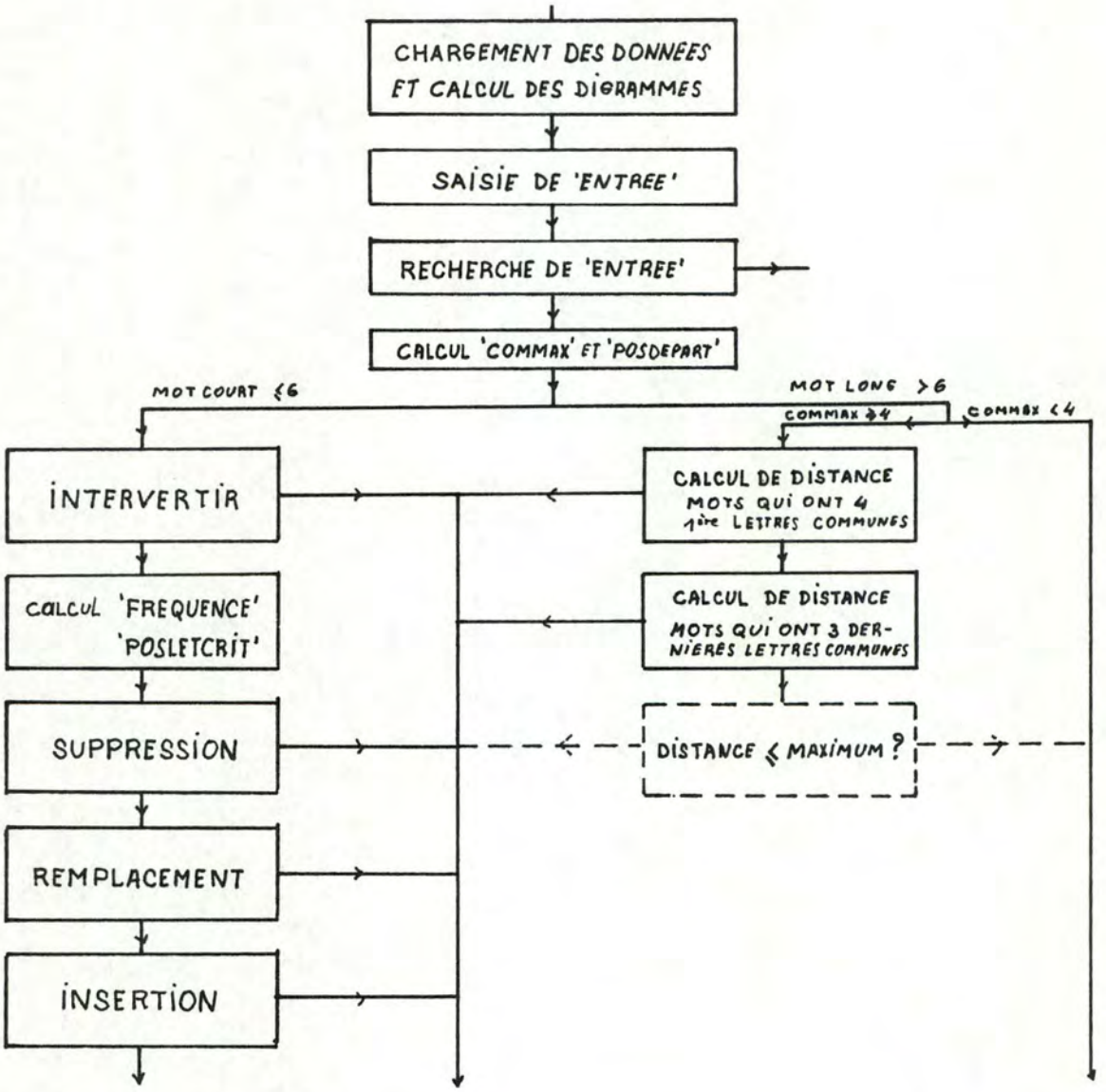
DICTIONNAIRE : file of string[20];

Fichier des mots de vocabulaire, classé par ordre alphabétique.

c. idée de solution, architecture du programme

Je vous ai bien eu, l'idée de solution elle est dans le mémoire à partir de la page 44.

Pour me faire pardonner, voici l'architecture du programme.



Le pointillé représente une extension très simple du programme. Actuellement, la procédure ESCAPE trouve toujours une solution. Il se peut que la distance entre le mot et la proposition de correction soit très grande. Pour ne pas proposer une ânerie, on peut ajouter un petit test qui refuse une proposition de correction si la distance est trop grande.

d. Spécifications des procédures

. Le programme principal

PROGRAMME CORMOT

arguments: ENTREE mot dont il faut éventuellement corriger l'orthographe
DICTIONNAIRE fichier des mots connus du programme

résultats: le programme nous propose 3 types de résultats

- 'orthographe correcte' si ENTREE existe dans le DICTIONNAIRE
- 'je ne connais pas ce mot'
  .si ENTREE compte compte moins de 7 lettres et que

    . la substitution d'une lettre par une autre
    . l'inversion de deux lettres
    . l'ajout d'une lettre
    . le retrait d'une lettre

  ne permettent pas de transformer ENTREE en un des mots du DICTIONNAIRE

  .si ENTREE compte plus de 6 lettres et que ni ses 4 premières ni ses 3 dernières ne sont communes à au moins un mot du DICTIONNAIRE

- le mot qui lui semble le plus probablement être celui que l'utilisateur à voulu écrire, l'idée de solution qui se trouve dans le mémoire vous précisera le 'qui lui semble'.

. Les procédures

Je choisis de vous les présenter dans l'ordre où elles apparaissent dans le programme.

PROCEDURE CHARGEMENT;

arguments : DICTIONNAIRE fichier des mots connus du programme

résultats : LISTMOT , DIMCAR
INDICE , DIMMOT
INDICETERM , DIMTERM

fonction : Charge chaque mot de DICTIONNAIRE en mémoire centrale par ordre alphabétique.

- Les mots seront enfilés bout à bout, caractère par caractère, dans LISTMOT séparés de '@'. Un mot supplémentaire 'ZZ' bidon sera crée*.

- A chaque mot correspond un élément dans INDICE, l'index de cet élément sera le numéro du mot par ordre alphabétique, la valeur de cet élément sera l'adresse du premier caractère du mot dans LISTMOT.

- Si le mot fait plus de 6 lettres, l'adresse dans LISTMOT de la troisième lettre avant la fin de ce mot sera chargée dans INDICETERM [i], i est le numéro d'ordre alphabétique de la terminaison du mot parmi les terminaisons des mots de plus de 6 lettres.

La procédure nous renvoie également le nombre d'éléments de chacun de ces trois tableaux.

* Pourquoi 'ZZ' ?

Il s'agit d'une astuce qui nous permet en toute généralité de connaitre la longueur d'un mot, sans exception pour le dernier.

Comment procédons-nous ? Ayez à l'esprit les spécifications de LISTMOT et INDICE. La longueur du ième mot est la différence entre l'adresse du 1er caractère du ième mot et du i+1 ème, moins un (le séparateur). Sans mot bidon, si i était le dernier mot comment aurions-nous défini le i+1 ème pour calculer la différence ?

PROCEDURE CORSTRING;

argument : -

résultat : CORSTR : TAB27
où TAB27 = packed array ['@'..'Z'] of string [1];

fonction : Construit une table de correspondance de types.
Si i est un caractère, CORSTR [i] est le même caractère mais dans une variable de type string [1].

Cette astuce est propre au Pascal UCSD qui n'accepte pas l'insertion d'un caractère de type char dans un string.

PROCEDURE CONSMATDIG;

arguments : LISTMOT, DIMCAR

résultat : DIGRAM

fonction : Constitue la matrice DIGRAM des fréquences des digramme que l'on trouve parmi les mots enfilés dans LISTMOT.

idée de solution :

Initialiser la matrice à 0
Pour i = 1 .. (DIMCAR-4)* faire

DIGRAM [LISTMOT i,LISTMOT i+1] :=DIGRAM [LISTMOT i,LISTMOTi+1] +

* on ne tient pas compte du mot bidon

PROCEDURE CHARBYTE;

arguments : SOURCE indice dans LISTMOT de la première lettre du mot à charger
LONGUEUR longueur en bytes du mot à charger, ici il s'agit aussi de la longueur en nombre de caractères car LISTMOT est compacté et donc un caractère égale un byte.

résultat : DESTINATION string où l'on charge le mot commençant à LISTMOT [SOURCE] et comptant LONGUEUR caractères

fonction : Extraire un mot de LISTMOT à partir de l'adresse SOURCE

motivation: Une question de performance nous a conduit à cette astuce.

La solution précédente était d'extraire lettre par lettre de LISTMOT le mot désiré. C'était trop lent.

La solution finalement adoptée nous permet d'employer une procédure Pascal UCSD de plus bas niveau qui ne travaille plus sur les variables mais bien sur les bytes qui les matérialisent.

Pour d'autres précisions, il faut consulter le manuel Pascal UCSD à propos de la procédure MOVELEFT.

PROCEDURE COMPARE;

arguments : ENTREE mot à comparer
FICHMOT mot de base auquel on compare

résultat : COMCOUR entier, nombre de lettres initiales communes entre ENTREE et FICHMOT

fonction : Calculer le nombre de lettres initiales communes entre ENTREE et FICHMOT.

PROCEDURE RECHERCHE;

argument : ENTREE mot que l'on recherche dans le dictionnaire

résultats : ORTHO booléen, vrai si ENTREE est dans le dictionnaire faux sinon
BI,BS adresses dans INDICE, si ORTHO est vrai le calcul (BI+BS)div2 donne le numéro d'ordre alphabétique de ENTREE; si ORTHO est faux BI et BS sont deux adresses consécutives entre lesquelles devrait se trouver ENTREE si elle existait.

fonction : Cherche si ENTREE est dans le dictionnaire, si oui ORTHO est mis à vrai sinon ORTHO est mis à faux. En outre le couple (min(BI,BS),max(BI,BS)) fournit le numéro d'ordre alphabétique qu'aurait ENTREE si on l'insérait dans le dictionnaire

idée de solution : La liste est longue de 280 mots. En séquentiel, le nombre moyen d'accès pour rechercher un mot donné est 140. Le nombre de recherches dans le programme, pour les mots courts, peut monter à plus de 60 dans les cas défavorables. Cela nous mène aux environs des 8500 accès. Par expérience je sais que cela coûte au moins, au moins 45sec! (une boucle de 1 .. 1000 qui exécute une simple addition à chaque tour prend environ 1sec)

La solution retenue est dès lors d'employer une recherche dichotomique, le nombre moyen d'accès sera ramené à 9. Le gain est appréciable n'est-ce pas ?

PROCEDURE CONTCARCOM;

arguments : BI,BS adresses consécutives entre lesquelles devrait se trouver ENTREE si on l'insérait dans le dictionnaire
ENTREE mot à analyser

résultats : COMMAX plus grand nombre de premières lettres que ENTREE ait en commun avec un mot du dictionnaire
POSDEPART numéro du mot qui possède le plus grand nombre de première lettres communes avec ENTREE.

fonction : Met dans COMMAX le plus grand nombre de lettres initiales que ENTREE possède en commun avec un mot du dictionnaire et renvoie dans POSDEPART le numéro du mot.

idée de solution : Comme BI et BS sont les indices des mots les plus proches par ordre alphabétique, nous allons seulement compter de ces deux derniers lequel a le plus de premières lettres communes Si BS = 0 POSDEPART égale BI, si BI = DIMMOT alors POSDEPART a la valeur de BS.

PROCEDURE CALPRODIGMOT;

arguments : ENTREE, COMMAX

résultats : FREQUENCE, DIMFREQ
FREQNUL
PROD, DIMPROD

fonction : Garnir FREQUENCE, DIMFREQ, FREQNUL, PROD, DIMPROD à partir des COMMAX+1 ou des COMMAX premières lettre de ENTREE.

idée de solution : Essayons d'être clair.

- Rappelons-nous que PROD

PROD[i] = FREQUENCE[i] X FREQUENCE[i+1] où

FREQUENCE[i] = DIGRAM[ENTREE[i-1], ENTREE[i]]

1ère question : Pour quelles valeurs de i, donc pour quels caractères ENTREE[i] faut-il calculer le produit des fréquences

Pour chacuns des caractères de ENTREE que nous allons essayer de remplacer ou devant lequel nous allons essayer d'insérer, c'est à dire les COMMAX+1 premiers caractères si ENTREE en possède au moins COMMAX+1

ex: dans le dictionnaire

. SALE
. SOLEIL
. SOT

le mot proposé

SOLTEIL

commax = 3

Selon nos hypothèses, cf mémoire p52, l'erreur se trouve parmi les 4 premiers caractères (ici, c'est le T qui est en trop). Pour connaitre la "chance" de chaque lettre d'être bonne, nous calculons le produit des fréquences pour les 4 premières lettres donc COMMAX+1*.

On calcule parfois le produit des fréquences pour seulement les COMMAX premiers caractères de ENTREE

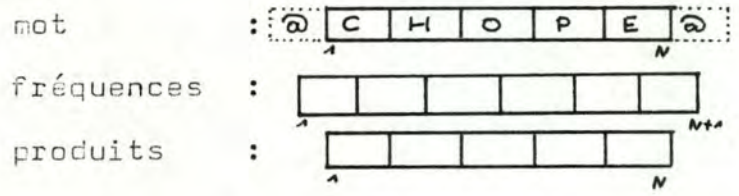
ex: Dans le même dictionnaire, le mot proposé est SOL.
Vous êtes d'accord, commax = 3.

Il serait difficile de traiter plus de 3 lettres dans ce cas, allez donc remplacer la 4ème lettre du mot SOL ! Nous calculerons donc le produit des fréquences pour 3 valeur, donc COMMAX* si COMMAX égale la longueur de ENTREE.

* S'il s'avérait qu'un des produits est nul, c'est que le premier des deux facteurs est nul (quand un facteur est nul, c'est que 1 des deux lettres du digramme est fausse et selon notre hypothèse "une seule faute pour les mots courts", nous ne calculons pas le second facteur). PROD ne contiendrait donc des valeurs que pour les lettres de ENTREE participant à ce 1er digramme nul.

2ème question : Connaissant le nombre de valeurs de PROD à calculer, combien cela fait-il de fréquences ?

Cela en fait N+1. Voyez le petit dessin ci-dessous.



3ème question :Comment référencer d'une manière générale le caractère '@' précédent le mot pour calculer le 1er digramme et comment référencer ENTREE N+1 qui n'existe pas nécessairement ?

En effectuant les calculs sur une variable de travail.

Si ENTREE N+1 n'est pas défini (longueur de ENTREE=N)

alors TRAVAIL:=concat('@',ENTREE,'@')
sinon TRAVAIL:=concat('@',ENTREE)

Ainsi nous aurons

dans TRAVAIL [1] le caractère qui précède ENTREE [1]
TRAVAIL [2] le caractère ENTREE [1]
TRAVAIL [N+1] le caractère ENTREE [N]
TRAVAIL [N+2] le carctère qui suit ENTREE [N] soit ENTREE [N+
soit '@'

L'algorithme se déroule donc en 2 phases

. Construire la variable de travail

. Calculer les N+1 fréquences et les N valeurs de PROD, éventuellement traiter le 1er digramme nul.

Les tableaux FREQUENCE et PROD se construisent en une seule boucle. Pour comprendre le test IF FREQUENCE[DIMFREQ] = 0 , consultez les spécifications de PROD et de FREQNUL p A.III.4.5.6

PROCEDURE RECHMINPROD;

arguments : PROD, DIMPROD, FREQNUL, ENTREE

résultats : POSLETCRIT : TAB6;
DIMPOSLETCRIT indice du plus grand élément de POSLETCRIT qui soit défini.

fonction : Garnir POSLETCRIT avec les indices de PROD . POSLETCRIT[1] est l'indice du plus petit élément de PROD, POSLETCRIT[DIMPOSLETCRIT] est l'indice du plus grand élément de PROD , POSLETCRIT[i] est l'indice du ième plus petit élément de PROD.

...Bon d'accord voilà l'exemple.

PROD : | 8 | 1 | 3 |
1 2 3

POSLETCRIT : | 2 | 3 | 1 |
1 2 3

idée de solution : Quelques explications seulement, ce n'est pas très difficile.

si FREQNUL=0
alors toutes les fréquences sont non nulles et aussi tous les éléments de PROD. La construction se fera selon une méthode de tri très générale que vous saisirez facilement.

si FREQNUL≠0 (le FREQNULième digramme est nul)

si FREQNUL=1
c'est le 1er digramme '@',ENTREE[1]qui est nul, la première lettre de ENTREE est surement fausse et on s'occupe plus des autres

alors POSLETCRIT[1] := 1
DIMPOSLETCRIT:= 1

sinon si (PROD[FREQNUL-1] = 0) et (PROD[FREQNUL] = 0)

alors un digramme est nul, ce n'est ni le premier ni le dernier. Il faudra traiter les 2 lettres qui y participent, garnissont POSLETCRIT de leurs références

```
                    POSLETCRIT [1] := FREQNUL-1
                    POSLETCRIT [2] := FREQNUL
                    DIMPOSLETCRIT:= 2

               sinon c'est le dernier digramme qui est nul
                     seule la dernière lettre nous intéresse

                    POSLETCRIT [1] := FREQNUL-1
                    DIMPOSLETCRIT:= 2 .
```

En somme, dans cette procédure nous sommes occupés à fabriquer le fil conducteur des tentatives de remplacement et d'insertion ultérieures.

Que l' on remplace ou qu'on insère, il faudra commencer par la lettre indicée par POSLETCRIT 1 , c'est celle dont la situatio entre ses voisine est la moins fréquente. Ensuite on traitera les lettres indicée par les autres éléments de POSLETCRIT jusque DIMPOSLETCRIT.

PROCEDURE SUPPRESSION;

arguments : ENTREE mot à analyser
DIMPOSLETCRIT
POSLETCRIT

résultats : ORTHO
ENTREE

fonction : Essayer de supprimer tour à tour les lettres de ENTREE, à chaque suppression, voir si le mot ainsi modifié est correctement écrit, si oui ORTHO est vrai et on arrête sinon on remet la lettre supprimée et on essaye avec une autre ORTHO reste faux.

PROCEDURE REMPLACLETTRE;

arguments : POSCRIT position dans ENTREE de la lettre à remplacer
ENTREE mot à traiter
CHOISIE lettre par laquelle on remplace
résultat : ENTREE dont la POSCRITième lettre a été remplacée par la lettre CHOISIE

PROCEDURE CLASREMP;

arguments : LETAV lettre qui précède celle que l'on va rempacer
LETAP lettre qui suit celle que l'on va remplacer

résultats : LETREMP TAB26
TAB26 = packed array[1..26] of string[1];

tableau reprennant les lettres qui peuvent se trouver entre LETAV et LETAP, par ordre décroissant des produits

DIGRAM[LETAV,lettre] X DIGRAM[lettre,LETAP]

COMPTLET nombre de lettres reprises dans LETREMP

fonction : Ordonner dans LETREMP les lettres de l'alphabet de façon telle que le produit associé à la lettre de LETREMP i soit que le produit associé à la lettre de LETREMP[i+1], et qu'aucune lettre de LETREMP ne soit associée à un produit nul. COMPTLET est le nombre de lettres retenues.

idée de solution :

Pour chaque lettre de l'alphabet, calculer le produit

produit = DIGRAM LETAV,lettre X DIGRAM lettre,LETAP

Il faut classer dans LETREMP les lettres associées à ces produits par ordre décroissant de ces produits.

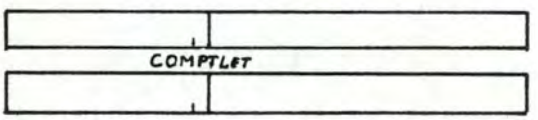
Une solution rapide consistera à classer une lettre dès qu'on aura calculé le produit.

Comment s'y prendre ?

J'ai imaginé un tableau de travail VALREMP. A tout moment, VALREMP[i] contient le produit associé à la lettre classée dans LETREMP[i]. VALREMP[i] ≤ VALREMP[i-1] pour i = 2 .. COMPTLET

Supposons que nous ayons traité les i-1 premiers produits.

A ce moment la situation est la suivante:



DERVAL = VALREMP[COMPTLET], c'est la plus petite valeur de produit rencontrée.
DERVAL>0

VALREMP[j] est le produit associé à la lettre LETREMP[j] pour j ≤ COMPTLET

COMPTLET est le nomdre de produits non nuls rencontrés.

Survient alors le ième produit que nous venons de calculer.

. Si ce produit est nul

On calcule le i+1ème produit sans changer la situation initiale.

. Si ce produit est $\leqslant$ DERVAL

```
DERVAL := produit
COMPTLET := COMPTLET+1
VALREMP COMPTLET := produit
LETREMP COMPTLET := lettre
```

. Si ce produit est $>$ DERVAL

Insérer produit dans VALREMP et lettre dans LETREMP.
Ne pas toucher à DERVAL.
COMPTLET := COMPTLET+1

Que signifie insérer dans ce cas ?

Cela consiste à :

. Rechercher j tel que VALREMP (j-1) $\geqslant$ produit et
VALREMP (j) $<$ produit

. Décaler vers la droite tous les éléments de VALREMP et LETREMP à partir de cette jème position pour faire une place libre en jème position de ces deux tableaux.

```
VALREMP [COMPTLET+1] := VALREMP [COMPTLET]
LETREMP [COMPTLET+1] := LETREMP [COMPTLET]

    .
    .
    .

VALREMP [j+1] := VALREMP [j]
LETREMP [j+1] := LETREMP [j]
```

. VALREMP [j] := produit
LETREMP [j] := lettre

La situation finale est la même que la situation initiale

VALREMP et LETREMP sont toujours triés
VALREMP [i] est toujours le produit associé à LETREMP [i]
DERVAL est toujours le plus petit produit rencontré
et est $>$ 0.

PROCEDURE LETAVAP;

arguments : ENTREE mot à traiter
POSCRIT position dans ENTREE de la lettre dont on cherche la précédente et la suivante
résultats : LETAV caractère qui précède ENTREE POSCRIT
LETAP caractère qui suit ENTREE POSCRIT

PROCEDURE REMPLACEMENT;

arguments : ENTREE mot à traiter
POSLETCRIT tableau des positions des lettres à traiter, classé par probabilités décroissantes que ces lettres soient fausses
DIMPOSLETCRIT longueur utile de POSLETCRIT

résultats : ENTREE mot éventuellement corrigé
ORTHO vrai si le mot est corrigé, faux sinon

fonction : Essayer de remplacer tour à tour les DIMPOSLETCRIT premières lettres de ENTREE. A chaque essai voir si ENTREE modifiée existe au dictionnaire si oui ORTHO est vrai et on arrête, si non on remet la lettre remplacée, ORTHO reste faux et on essaye avec une autre

idée de solution :

- Essayer de remplacer les lettres de ENTREE dans l'ordre qui est donné par POSLETCRIT
- Pour savoir par quelle lettre remplacer celle qu'on enlève, déterminer les caractères qui la précède et la suive et employer la procédure CLASLETREMP.

PROCEDURE CLASPOSINS;

arguments : FREQNUL = 0 si aucun digramme n'a de fréquence nulle
= i si le ième digramme est nul
FREQUENCE tableau des commax+1 premiers digrammes de ENTREE
DIMFREQ longueur de FREQUENCE

résultat : POSDIGCRIT tableau d'entiers reprennant les positions des digrammes de FREQUENCE classées par ordre croissant des valeurs de digramme correspondantes
DIMPOSDIGCRIT longueur de POSDIGCRIT

fonction : garnit le tableau POSDIGCRIT des positions des éléments du tableau FREQUENCE en le classant par ordre croissant des valeurs de digramme correspondantes

PROCEDURE LETGAUDROI;

arguments : TRAVAIL mot sur lequel on travaille
POSCRIT position du digramme concerné dans TRAVAIL

résultats : LETGAU lettre de gauche du POSCRITième digramme de TRAVAIL
LETDROI lettre de droite du POSCRITième digramme de TRAVAIL

fonction : Mettre dans LETGAU la lettre gauche et dans LETDROI la lettre droite du POSCRITième digramme de TRAVAIL.

PROCEDURE INSERTION;

arguments : ENTREE mot à traiter
FREQUENCE tableau des commax+1 premiers digrammes de ENTREE
DIMFREQ longueur de FREQUENCE
FREQNUL = 0 si aucun digramme n'a de fréquence nulle
= i si le ième digramme de ENTREE a une fréquence nulle

résultats : ENTREE mot éventuellement corrigé
ORTHO vrai si le mot est corrigé
faux sinon

fonction : Essaye de corriger le mot ENTREE par insertion d'une lettre parmi les commax+1 premières.
Cette procédure détermine l'emplacement opportun et les lettres de remplacement.

PROCEDURE INTERVERTIR;

arguments : ENTREE mot à traiter
COMMAX plus grand nombre de lettres initiales que ENTREE à en commun avec un mot du dictionnaire

résultats : ENTREE mot éventuellement corrigé
ORTHO vrai si le mot est corrigé
faux sinon

fonction : Essaye de corriger le mot ENTREE en inversant deux lettres voisines parmi les commax premières lettres.

PROCEDURE ESCAPE;

arguments : ENTREE mot à traiter
COMMAX plus grand nombre de lettres initiales communes que possède ENTREE avec un mot du dictionnaire
POSDEPART index dans INDICE du mot qui possède COMMAX lettres communes avec ENTREE

résultats : ENTREE éventuellement corrigé
ORTHO vrai si le mot est corrigé, faux sinon

fonction : Recherche parmi les mots de plus de 6 lettres ayant au moins leurs 4 premières lettres communes avec ENTREE ou leur 3 dernières lettres communes, le mot qui ressemble le plus à ENTREE. 'qui ressemble le plus', c'est l'application de l'algorithme de distance décrit dans le mémoire.

idée de solution : pour les mots de plus de 6 lettres seulement !

- faire la liste des mots ayant au moins 4 premières lettres communes avec ENTREE

  appliquer l'algorithme de distance entre ENTREE et cette liste

- si on n'a rien trouvé

  faire la liste des mots ayant leurs 3 dernières lettres communes ave ENTREE

  appliquer l'algorithme de distance entre ENTREE et cette liste

PROCEDURE DISTANCE;

arguments : ENTREE 1er mot
CHAINE 2ème mot
DISTMIN plus petite distance déja atteinte entre 2 mots

résultat : DIST distance entre les 2 chaines ENTREE et CHAINE
DISTMIN = min (DIST,DISTMIN)

fonction : Calculer la distance entre ENTREE et CHAINE pour autant que cette distance ne devienne pas plus grande ou égale à DISTMIN

idée de solution :

voici nos deux mots

ENTREE E $_i$

CHAINE C $_j$

que par facilité j'appelerai E ou C,

supposons que nous sommes au ième élément de E
jème élément de C

nous allons voir successivement, tant que les réponses sont non s

E [i] = C [j] ?

si oui i:=i+1
j:=j+1

si non y a-t-il inversion entre E [i] et E [i+1] ?

si oui distance:=distance+1
i:=i+2
j:=j+2

si non y a-t-il substitution ?

si oui distance:=distance+1
i:=i+2
j:=j+2

si non y a-t-il ajout ?

si oui distance:=distance+1
long:=long+1
i:=i+2
j:=j+1

si non y a-t-il oubli ?

si oui distance:=distance+1
long:=long-1
i:=i+1
j:=j+2

si non suis-je au bout d'une des chai

si oui la faute n'est pas
analysée mais
elle existe
distance:=distance+1

si non la faute porte sur 2
lettres consécutives
distance:=distance+2

i:=i+1
j:=j+1
(j'avance de 1 et pas de 2 pour qu
ça coûte très cher si les 2 pro-
chaines lettres sont encore ≠

au début, on initialisera i et j à 1, distance à 0, long est la longueur de CHAINE

une remarque sur long : vous constatez que je pénalise un mot quand je trouve qu'il y a une lettre en trop ou trop peu, l'emploi de long me permet de ne plus le pénaliser à la fin quand je mesure le longueurs. Exemple à la page, ici je dérape.

Ex . j'écris ERBLE

pour ERABLE

Selon mon algorithme, je constate que E[3] ≠ C[3], que E[4] ≠ C[3], et que E[3] = C[4].

Je déduis qu'il manque une lettre avant E[3] et fièrement je comptabilise distance:=distance+1.

J'arrive à la fin de mon 1er mot, et dans ma tête à moi je me convaincs très logiquement 'si un mot est plus court que l'autre, c'est qu'il manque les lettres' et fièrement je comptabilise distance:=distance+ DIFFERENCE DE LONGUEUR

Je ne sais plus comment je me suis aperçu que j'avais pénalis deux fois la même faute, je ne sais plus.

PROCEDURE INTERCHANGER;

arguments : - ENTREE[i], CHAINE[j]

résultat : RESULTAT booléen **vrai** s'il y a eu inversion
faux sinon

fonction : vérifier si ENTREE[i+1] = CHAINE[i]
et si ENTREE[i] = CHAINE[i+1]
les 2 conditions réunies signifiant l'inversion

PROCEDURE SUBSTITUER;

arguments : - ENTREE[i], CHAINE[j]

résultat : RESULTAT booléen vrai s'il y a eu substitution
faux sinon

fonction : vérifier si ENTREE[i+1] = CHAINE[i+1], auquel cas il y aurait substitution

PROCEDURE AJOUTER;

arguments : - ENTREE[i], CHAINE[j]

résultat : RESULTAT booléen vrai s'il y a ajout d'une lettre
faux sinon

fonction : vérifier si ENTREE[i+1] = CHAINE[i], auquel cas il y aurait eu ajout d'une lettre

PROCEDURE OUBLIER;

arguments : - ENTREE[i], CHAINE[j]

résultat : RESULTAT booléen vrai s'il manque une lettre
faux sinon

fonction : Vérifier si ENTREE[i] = CHAINE[i+1], auquel cas il manquerait une lettre.

PROCEDURE LISTCANDIDAT;

arguments : POSDEPART adresse d'un mot qui a COMMAX premières lettres communes avec ENTREE

COMMAX nombre maximum de lettres initiales communes que ENTREE a en commun avec un mot du dictionnaire

ENTREE mot à corriger

résultats : DISTMIN distance la plus petite qu'il puisse y avoir entre ENTREE et un des mots du dictionnaire ayant COMMAX lettres initiales communes avec ENTREE

INTER mot ayant ses COMMAX premières lettres communes avec ENTREE et distant de celui-ci de DISTMIN

fonction : Recherche dans le dictionnaire les mots ayant leurs COMMAX premières lettres communes avec ENTREE et met celui qui est le plus proche dans INTER, la distance entre ces deux mots étant chargée dans DISTMIN.

idée de solution :

Partir de POSDEPART, profiter de l'ordre alphabétique et analyser tous les mots avant POSDEPART ayant les fameuses commax lettres communes et puis analyser les mots après POSDEPART.

PROCEDURE LISTSUFFCANDI;

arguments : SUFFIXE ce sont les trois dernière lettres de ENTREE
POSDEPART adresse dans INDICETERM d'une terminaison égale à SUFFIXE
DISTMIN distance minimum déja trouvée entre ENTREE et un mot du dictionnaire
ENTREE mot à corriger

résultats : DISTMIN distance la plus petite qu'il puisse y avoir entre ENTREE et un mot du dictionnaire ayant

leurs COMMAX premières lettres ou leurs trois dernières communes

INTER mot qui répond à la condition de plus petite distance

fonction : Rechercher dans le dictionnaire le mot ayant ses trois dernières lettres communes avec ENTREE et distant de celui-ci d'une valeur inférieure à DISTMIN.

L'idée de solution est la même que pour LISTCANDIDAT, analyser d'abord les suffixes d'un côté de POSDEPART et puis de l'autre.

ANNEXE A.IV

SPECIFICATIONS DES PROGRAMMES DISPONIBLES POUR GERER LES DONNEES

- GESDICTIO
- CREFICH
- TERM
- CREETERM
- GERSEMA
- CRESEM
- CLASSE

SPECIFICATIONS DES PROGRAMMES DISPONIBLES POUR GERER LES DONNEES

a. Introduction
b. Présentation des programmes

1. GESDICTIO
2. CREEFICH
3. TERM
4. CREETERM
5. GERSEMA , CRESEM
6. CLASSE

Pour chacun de ces programmes, je présenterai

- Les commandes disponibles
- Leur mode d'emploi
- L'architecture du programme et les spécifications des procédures compliquées*
- Eventuellement des remarques sur ce qu'il faudrait implémenter pour réaliser certaines actions sur les données

* Parmi l'ensemble des procédures, la majorité réalise des sorties sur écran ou des extractions de fichier. Ces procédures ne seront pas spécifiées ici mais commentées dans le texte du programme.

a. Introduction

Les données à gérer appartiennent aux fichiers suivants, définis dans les spécifications du programme GENERATION cf p .

- FICHPREFIXES (#5:TEST2.DATA)
- FSTXNCOM (#5:STXNCOM.DATA)
- FSTXADJ (#5:STXADJ.DATA)
- FESTXVB (#5:STXEVERBE)
- FISSTXVB (#5:STXISVERBE)
- FSSTXVB (#5:STXSVERBE)

qui sont gérés par les programmes GESDICTIO et CREFICH

- TERM (#5:BESCH.DATA)

qui est géré par les programmes TERM et CREBESCH

. FSEMEVB (#5:SEMEVB.DATA)
. FSEMISVB (#5:SEMISVB.DATA)
. FSEMSVB (#5:SEMSVB.DATA)
. FSEMNCOM (#5:SEMNCOM.DATA)
. FSEMADJ (#5:SEMADJ.DATA)

qui sont gérés par le programme GERSEMA

. FCLASSES (#5:CLASSES.DATA)

qui est géré par le programme CLASSES

Soit 13 fichiers de natures différentes et dépendants les uns autres. Je voudrais vous rappeler que l'objectif de mon mémoire était de développer une application EAO. Cette application s'est vite orientée vers la nécessité de générer des phrases pour donner aux enfants des exemples de cas d'accord du verbe.

Pour générer des phrases, il faut des mots. Beaucoup de mots. Il faut toutes les sortes de mots, non là j'exagère il ne les faut pas toutes, seulement quelques unes:des verbes, des noms et quelques fifrelins. Vous avez lu le mémoire, vous savez.

Ceci pour dire qu'il me fallait garnir une base de données de choses assez diverses.

Pour créer chacun des fichiers et les remplir, je me suis écrit les quelques programmes de gestion de données nécessaires. J'en ai fait le MINIMUM. Ce n'est pas par paresse, ni par négligence. C'est par manque de temps.

Mon objectif, c'est rentrer au plus vite mes données pour tester au plus tôt la génération de phrases. Si je traine, je m'enlise, je meurs. Gérer une base de données, ce n'est pas mon mémoire, je préfère avancer dans l'application EAO, il me faut un résultat cette année, il faut attirer le client pour que le projet continue.

Il va de soi que si l'application terminée l'une ou l'autre perssonne désirait l'acquérir, je refuserais de lui céder la gestion de base de données dans cet état. Je serais vraiment honteux et puis ce serait malhonnète.

En bref je vous dirai que ces programmes de gestion sont écrits pour répondre à mes besoins.

- Beaucoup d'ajouts en une fois, les éléments étant entrés dans l'ordre où ils doivent se trouver dans la base.

- Peu de modifications

Les besoins d'une autre personnes seraient par contre

- Beaucoup d'insertions

- Grosses modifications ex refaire des classes, et en général modifier tout ce qui provient d'un choix subjectif.

b. Présentation des programmes

avertissement : Les noms de types et de variables employés ici ont été spécifiés pour le programme GENERATION p**A.I**.

1. Programme GESDICTIO

- Les commandes

Ce programme gère les fichiers FICHPREFIXES, FSTXADJ, FSTXNCOM, FESTXVB, FISSTXVB, FSSTXVB.

Il permet de consulter les mots d'une nature déterminée
d'ajouter un nouveau mot
de supprimer un mot
de modifier un mot (pas implémenté)
d'imprimer les mots d'une nature déterminée

Lorsque je parle de mots, vous devez vous rappeler que ce mot est divisé en un radical qui se trouve dans FICHPREFIXES et une partie variable qui se trouve dans un des autres fichiers. Vous aurez plus de détails en vous référant au mémoire p59.

- Mode d'emploi

Le programme vous aide, il vous faudra répondre aux questions posées à l'écran.

Après une minute de patience, le temps de charger la mémoire centrale, s'affichera le menu principal.

Vous choisirez votre orientation dans ce menu, à savoir

- consultation
- ajout
- modification (pas implémenté)
- suppression
- impression

Si vous choisissez 'consultation'

Le programme vous demande alors ce que vous voulez consulter.

1. liste des radicaux
2. liste des noms propres féminins
3. liste des noms propres masculins
4. liste des noms communs et leurs caractéristiques
5. liste des verbes en 'E'
6. liste des verbes en 'IS'
7. liste des verbes en 'S'
8. liste des adjectifs
9. liste des adverbes

Pour la liste des radicaux

Le programme vous donnera la liste alphabétique des radicaux, avec pour chaque radical:

- le numéro de radical
- la nature du mot qui possède ce radical
- le numéro d'ordre de ce radical parmi les mots de même nature

Pour la liste des noms propres féminins

Le programme vous donnera la liste alphabétique des noms propres féminins avec pour chaque nom:

- le numéro d'ordre de ce nom parmi l'ensemble des radicaux

Pour la liste des noms propres masculins

Le programme vous donnera la liste alphabétique des noms propres masculins avec pour chaque nom:

- le numéro d'ordre de ce nom parmi l'ensemble des radicaux

Pour la liste des noms communs

Le programme vous donnera la liste alphabétique des noms communs avec pour chaque nom:

- le numéro de ce nom commun
- les différents accords de ce nom commun
- les caractéristiques de ce nom commun
- le numéro d'ordre de ce nom parmi l'ensemble des radicaux

Pour la liste des verbes en 'E'

Le programme vous donnera la liste alphabétique des verbes en 'E' à l'infinitif avec pour chaque verbe

- le numéro du verbe en 'E'
- le numéro d'ordre de son groupe BESCHERELLE
- le numéro d'ordre de ce verbe parmi l'ensemble des radicaux

Pour la liste des verbes en 'IS'

Le programme vous donnera la liste alphabétique des verbes en 'IS' à l'infinitif avec pour chaque verbe:

. le numéro du verbe en 'IS'
. le numéro d'ordre de son groupe BESCHERELLE
. lenuméro d'ordre de ce verbe parmi l'ensemble des radicaux

Pour la liste des verbes en 'IS'

Le programme vous donnera la liste alphabétique des verbes en 'S' à l'infinitif avec pour chaque verbe:

. le numéro du verbe en 'S'
. le numéro d'ordre de son groupe BESCHERELLE
. le numéro d'ordre de ce verbe parmi l'ensemble des radicaux

Pour la liste des adjectifs

Le programme vous donnera la liste alphabétique des adjectifs avec pour chaque adjectif:

. le numéro de l'adjectif
. les différents accords de cet adjectif
. le numéro d'ordre de ce nom parmi l'ensemble des radicaux

Pour la liste des adverbes

Le programme vous donnera la liste alphabétique des adverbes avec pour chaque adverbe:

. le numéro d'adverbe
. le numéro d'ordre de ce nom parmi l'ensemble des radicaux

La liste des éléments demandés se déroule sans s'arrêter. Pour interrompre le défilement, il est possible d'employer la commande CTRL/S.
La liste s'arrête sur le dernier élément, un message à l'écran vous dira comment repartir.
Apres la consultation, le programme revient au menu principal.

<u>Si vous avez choisi 'ajout'</u>

Alors le programme vous demande quelle est la nature du mot à ajouter.

- verbe
- nom propre
- nom commun
- adjectif qualificatif
- adverbe
- terminé si vous ne voulez pas ou plus ajouter

Dès que vous avez répondu, le programme vous demande le radical du nouveau mot (je rappelle que le radical est la partie invariable d'un mot).

Le programme effectue alors une vérification d'où il ressort que soit le radical existe déjà, dans ce cas il vous sera proposé de refaire l'ajout, soit le radical n'existe pas et le programme vous demande alors de l'information supplémentaire propre à la nature du nouveau mot.

Si c'est un verbe:

vous donnerez
- sa terminaisons infinitive
- le type de groupe E IS S de ce verbe
- le numéro d'ordre de sa conjugaison BESCHERELLE

Si c'est un nom propre:

vous donnerez
- son genre féminin ou masculin

Si c'est un nom commun:

vous donnerez
- sa terminaison fém sing si elle existe
- sa terminaison fém plur si elle existe
- sa terminaison masc sing si elle existe
- sa terminaison masc plur si elle existe
- s'il admet au moins un adjectif 'devant'
- s'il admet au moins un adjectif 'derrière'

Si c'est un adjectif qualificatif:

vous donnerez
- sa terminaison fém sing si elle existe
- sa terminaison fém plur si elle existe
- sa terminaison masc sing si elle existe
- sa terminaison masc plur si elle existe

Si c'est un adverbe:

vous ne donnerez rien du tout, le programme vous dira seulement que tout s'est bien passé.

Lorsque toutes les informations concernant le mot ajouté ont été donnée, le programme vous propose d'ajouter un autre mot, vous pouvez refuser en répondant '6' terminé.

Si vous avez terminé d'ajouter et que vous avez ajouté au moins un mot, le programme sauve votre travail sur disquette et vous en avertit, quelques secondes de patience seront nécessaires.

Le programme revient alors au menu principal.

Si vous n'avez rien ajouté, le programme revient au menu principal sans sauvetage.

Si vous n'avez pas fini d'ajouter, le processus recommence, (quelle est la nature du mot à ajouter etc...)

Si vous avez choisi 'supprimer'

Le programme vous demande le radical du mot à supprimer.

Si ce radical n'existe pas, il y a erreur et le programme vous redemande d'écrire le radical.

Si ce radical existe, le programme vous demande si vous voulez vraiment supprimer le mot pour éviter une suppression par erreur.

Si oui le programme supprime et vous demande s'il y en a d'autres à supprimer.
Si non le programme vous demande s'il y en a d'autres à supprimer.

S'il n y a plus rien à supprimer, le programme sauve la nouvelle version de la base de données sur disquette et rejoint le menu principal.

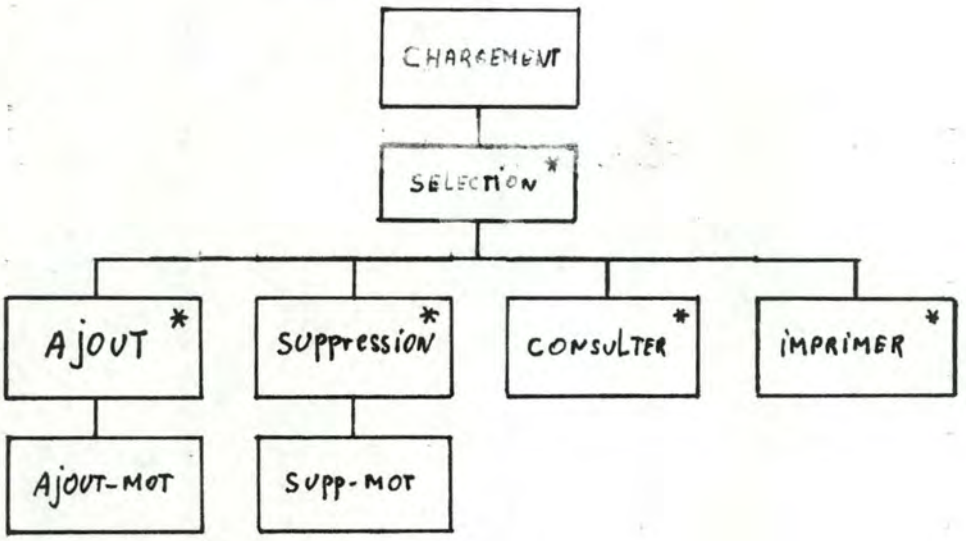
S'il y a encore des mots à supprimer le processus recommence.

Si vous avez choisi 'imprimer'

C'est exactement comme pour une consultation, mais la liste sort sur papier. Vérifiez que l'imprimante est branchée.

c. Spécifications des procédures compliquées

1. Architecture du programme



Les deux traitements difficiles à comprendre sont AJOUT et SUPPRESSION.

Avant de vous en parler, je vous touche un mot de CONSULTATION et de IMPRESSION.

Ces deux traitements ont exactement la même structure. Seul change le support où s'inscrivent les résultats: l'écran pour le premier, le papier pour le second.

algorithme de ces traitements :

```
tant que pas fini de consulter ou d'imprimer  faire

   début

       affichage d'un menu et saisie du choix de l'élément à
       consulter ou imprimer;

       parcourir le tableau correspondant à la nature de cet
       élément et sortir les informations demandées en soignant
       la présentation;

   fin;
```

Je rappelle que toutes les informations concernant l'organisation les types de données et les variables de la base de données ont été données dans le mémoire p54 et en annexe p.A1. dans les spécifications du programme GENERATTON.

Traitement AJOUT

algoritnme de ce traitement :

```
tant que pas fini d'ajouter  faire
   début
        demander la nature du mot à ajouter;
        demander le radical du mot à ajouter;
        si ce radical existe déja
           alors erreur on recommence l'ajout
           sinon
               début
                  grille de saisie d'informations selon la nature
                  du mot à ajouter;
                  ajout du mot dans les tableaux de la mémoire
                  centrale
               fin
   fin;
si il y a eu au moins un ajout
   alors  sauver la nouvelle version de la base sur disquette
   sinon  rien
```

La procédure difficile de ce traitement est celle qui ajoute le mot dans les tableaux de la mémoire centrale.

PROCEDURE AJOUTMOT;

arguments : MOT : radical du mot à ajouter
INDAP : numéro alphabétique de ce mot parmi l'ensemble des radicaux, soit l'adresse où il faut insérer dans INDICE
TERNC : informations concernant la terminaison du mot et ses caractéristiques si le mot à ajouter est un nom commun
TERADJ : informations concernant la terminaison du mot si c'est un adjectif
TEREV,TERSV,TERISV : informations concernant la terminaison infinitive et le le numéro d'ordre du groupe BESCHERELLE si c'est un verbe en E,S,IS

résultats :

LISTMOT

pour i ⩾ INDICE [INDAP].ADRESSE + longueur de MOT + 1

LISTMOT[i]:= LISTMOT[i-longueur de MOT-1];

LISTMOT [INDICE [INDAP].ADRESSE] := 1ere lettre de MOT

.
.
.

LISTMOT [INDICE [INDAP].ADRESSE +N-1]:= Nème lettre de MOT

nbrecar = nbrecar + longueur (MOT) + 1

INDICE

INDICE[i] varie pour i⩾INDAP

si i=INDAP

INDICE[i].ADRESSE rien à changer
INDICE[i].NAT = MOT.NAT
INDICE[i].NUMPARNAT = numéro d'ordre du nouveau mot parmi les autres mots de même nature

si i>INDAP

nbremot = nbremot + 1
INDICE[i].ADRESSE = INDICE[i-1].ADRESSE + longueur de MOT + 1
INDICE[i].NUMPARNAT = INDICE[i-1].NUMPARNAT
si MOT.NAT ≠ INDICE[i].NAT

= INDICE[i-1].NUMPARNAT + 1 sinon
INDICE[i].NAT = INDICE[i-1].NAT

pour tous les autres tableaux

pour i⩾INDAP

TNCOM[i].ADIND = TNCOM[i].ADIND + 1
TMNPROP[i] = TMNPROP[i]+ 1
TFNPROP[i] = TFNPROP[i]+ 1
TADV[i] = TADV[i] + 1
TADJQ[i].ADIND = TADJQ[i].ADIND + 1
TEVERBE[i].ADIND = TEVERBE[i].ADIND + 1
TISVERBE[i].ADIND = TISVERBE[i].ADIND + 1
TSVERBE[i].ACIND = TSVERBE[i].ADIND + 1

seulement pour le tableau correspondant à la nature du mot

NUM* = NUM* + 1

pour i = INDICE[INDAP].NUMPARNAT ... NUM*

TABLE*[i+1]= TABLE*[i];

TABLE* INDICE INDAP .NUMPARNAT = INDAP
TABLE* INDICE INDAP .NUMPARNAT = reste éventuel de la descriptio
(TERNC, TERADJ, TEREV, TERISV
TERSV)

.

Traitement SUPPRESSION

algorithme de ce traitement

tant que pas fini de supprimer faire

début

demander le radical du mot à supprimer;

si ce radical n'existe pas

alors erreur on recommence la suppression

sinon
début

demander si il faut vraiment supprimer;

si oui supprimer le mot
si non ne pas supprimer;

demander s'il y en a d'autre à supprimer

fin

fin;

si il y a eu au moins une suppression

alors sauver la nouvelle version de la base sur disquette
sinon rien

PROCEDURE SUPPRESMOT;

arguments : MOT radical du mot à supprimer
INDMOT indice de MOT dans INDICE

résultats :

```
LISTMOT

 pour i ≥ INDICE[INDMOT].ADRESSE

 LISTMOT[i] = LISTMOT[i+1];

 NBRCAR = NBRCAR - longueur de MOT - 1;


INDICE

 NBRMOT = NBRMOT-1;
 pour i INDMOT+1 .. NBRMOT
 INDICE[i].ADRESSE = INDICE[i].ADRESSE - longueur de MOT - 1

 pour i=INDMOT .. NBREMOT
 INDICE[i].* = INDICE[i+1].*
```

* = tous les champs

pour tous les autres tableaux

```
 pour i ≥ INDMOT

    TNCOM[i].ADIND = TNCOM[i+1].ADIND
    TMNPROP[i] = TMNPROP[i+1]
    TFNPROP[i] = TFNPROP[i+1]
    TADV[i] = TADV[i+1]
    TADJQ[i].ADIND = TADJQ[i+1].ADIND
    TEVERBE[i].ADIND = TEVERBE[i+1].ADIND
    TISVERBE[i].ADIND = TISVERBE[i+1].ADIND
    TSVERBE[i].ADIND = TSVERBE[i+1].ADIND
```

seulement pour le tableau correspondant à la nature du mot

```
 NUM* = NUM*-1

 pour i = INDICE INDMOT .NUMPARNAT .. NUM*

      TABLE[i]= TABLE[i+1]
```

Les procédures ajout et suppression étaient les seules qui pouvaient vous poser des problèmes.

En fait la grosse difficulté est de bien s'immprégner de la structure de la base de données, décrite dans le mémoire et des types de données et variables employés qui sont spécifiés pour le programme GENERATION en annexe p .

2. Programme CREFICH

Attention DANGER !

Je ne plaisante pas, écoute, si tu exécutes ce programme, tu tues, tu ruines, tu anéantis des heures de patience, les heures nécessaires à entrer 280 mots et leurs descriptions dans la base de donnée.

Tu a compris que ce programme initialise en écriture les fichiers FICHPREFIXES, FSTXNCOM, FSTXADJQ, FESTXVB, FISSTXVB, FSSTXVB.

Initialise en écriture, ça veut dire tout écrabouiller pour pouvoir en recopier un nouveau par dessus.

Si vous avez élaboré un nouveau dictionnaire c'est très bien. Entrez-le seulement. Je suis content que quelqu'un s'intéresse à ce travail. Mais s'il vous plait faites un back up de l'ancien dictionnaire, c'est si prudent ...

Pour le mode d'emploi, c'est tout simple tout est à l'écran.

Pour implémenter c'est aussi simple un REWRITE du fichier suivi d'un CLOSE

3. Programme TERM

a. Les commandes

Le programme propose le minimum nécessaire à savoir :

- ajouter un groupe de conjugaison
- modifier une terminaison dans un groupe de conjugaison
- consulter un temps dans un groupe de conjugaison
- supprimer un groupe de conjugaison.

b. Mode d'emploi

Le programme vous aide. Il vous propose d'abord un menu principal dans lequel vous choisirez votre orientation.

- ajouter
- consulter
- modifier
- supprimer

Si vous voulez ajouter

Le programme vous demandera le numéro d'ordre du groupe à ajouter. Ce numéro sera plus grand que zéro et plus petit ou égal au nombre de groupes existants + 1.

Le programme vous demandera alors les différentes terminaisons 1ere , 2ème , 3ème pers sing et 1ère, 2ème , 3ème pers plur pour les temps présent, imparfait, passé simple, futur simple et enfin il vous demandera la terminaison participe passé.

Cela fait 25 terminaisons.

Ayant fini d'ajouter un groupe le programme vous demande si vous voulez en ajouter un autre.

Si oui, le programme sauve toujours le groupe nouveau et vous aide à ajouter le prochain.

Si non, le programme sauve le groupe tout neuf et rejoint le menu principal.

Si vous voulez modifier une terminaison

Le programme vous demandera

- numéro du groupe à modifier plus grand que zéro et plus petit ou égal au nombre maximum de groupes existants
- numéro de temps dont il faut modifier une terminaison

  1 présent
  2 imparfait
  3 passé simple
  4 futur simple
  5 participe passé

- s'il ne s'agit pas du participe passé le programme vous demande la personne à modifier

  1 1ère sing
  2 2ème sing
  3 3ème sing
  4 1ère plur
  5 2ème plur
  6 3ème plur

Le programme vous rappellle alors l'ancienne version et

et vous demandera la nouvelle terminaison.

Immédiatement il sauvera la modification sur disquette et vous demandera s'il ya d'autre modification.

Il vous demandera alors s'il y a d'autres modifications à effectuer si oui il vous aide à les faire, sinon il rejoint le menu principal.

Si vous voulez consulter un temps d'un groupe de conjugaison

Le programme vous demandera le numéro du groupe de conjugaison que vous voulez consulter.

Il vous demandera ensuite le numéro du temps à consulter.

Il vous présente alors les terminaisons du temps et du groupe demandé.

Le programme vous demandera ensuite si vous voulez encore consulter si oui vous consulterez , si non le programme revient au menu principal.

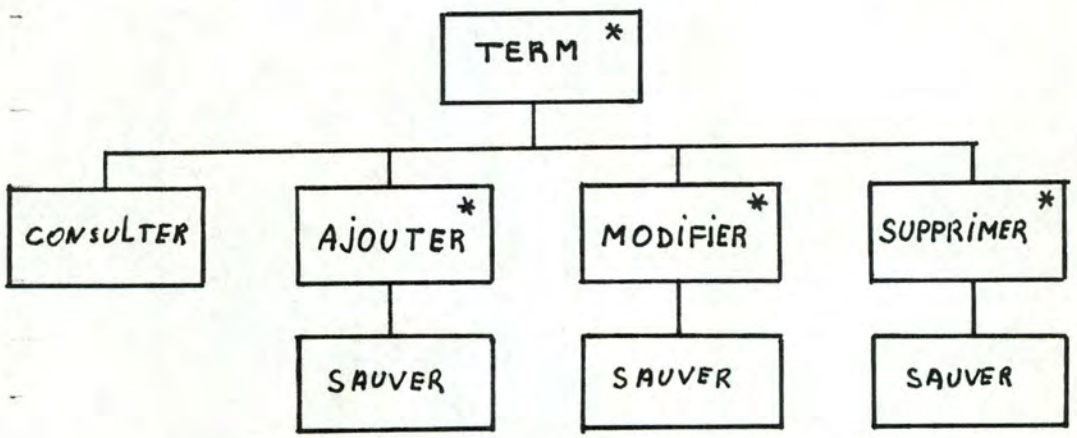
Si vous voulez supprimer

Le programme vous demandera le numéro du groupe à supprimer.

Il le supprime et sauve immédiatement la nouvelle version des données sur disquette.

Il vous demande ensuite si vous voulez supprimer un autre groupe. Si oui il recommence le processus de suppression, si non il revient au menu principal.

c. Architecture du programme et spécifications des procédures



Spécifications des procédures

Toutes les variables, tous les types de données onte été spécifiés pour le programme GENERATION p .

Quelques procédures ont aussi été spécifiées dans ce programme: ce sont CHARBESCH, ACCEDE, CHARGTERM.

PROCEDURE SAUVTERM;

arguments : - INDIRECT, TERMINAISON

résultats : le fichier TERM

fonction : sauver l'ensemble des terminaisons dans le fichier TERM

idée de solution :

Le principe est le même que pour le chargement, mais dans l'autre sens. Ben oui quoi!

Il s'agit de reconstituer les groupes champs par champs et de les recopier au fur et à mesure dans le fichier TERM.

Pour reconstituer un groupe, c'est simple:

Les 6 premières terminaisons remplissent le champ GROUPE.PRESENT i pour i= 1..6

Les terminaisons 7..12 remplissent le champ GROUPE.IMPARFAIT i pour i= 7..12

Les terminaisons 13..18 remplissent le champ GROUPE.PASSESIMPLE i pour i= 13..18

Les terminaisons 19..24 remplissent le champ GROUPE.FUTURSIMPLE i pour i= 19..24

La terminaison 25 remplit le champ GROUPE.PARTPASS

PROCEDURE AJOUTER;

arguments : -

résultats : INDIRECT, TERMINAISONS, le fichier TERM

fonction : Ajputer dans INDIRECT, TERMINAISON et lefichier TERM les nouvelles terminaisons entrées au clavier.

PROCEDURE LITTERM;

argument : ADINDIR adresse où l'on doit insérer la référence de la terminaison

résultats: la terminaison est lue et insérée dans TERMINAISON INDIRECT ADINDIR est la référence de cette terminaison, INDIRECT contient une référence de plus

fonction : Lire au clavier une terminaison et l'insérer dans TERMINAISON à l'adresse INDIRECT ADINDIR

idée de solution :

La terminaison doit s'insérer dans TERMINAISON à partir de l'adresse INDIERCT ADINDIR .

Pour cela on va décaler TERMINAISON de (longueur de la terminaison) positions vers la droite et insérer la terminaison dans ce trou.

Bien sur NBTER:=NBTER+1

Ayant inséré cette terminaison, il faut mettre à jour les références dans INDIRECT. On va donc décaler de une position vers la droite tous les éléments de INDIRECT à partir de ADINDIR et ajouter à chaque référence déplacée la longueur de terminaison.

ADR:=ADR+1

ADINDIR:=ADINDIR+1

PROCEDURE CREETEMPS;

argument : ADINDIR adresse dans INDIRECT où s'insère la référence de la 1ère terminaison du temps à ajouter

résultat : 6 terminaisons sont saisies et insérées
ADIDIR est l'adresse où l'on insérera la prochaine terminaison du même groupe

PROCEDURE CREEPARTPASS;

argument: ADINDIR adresse dans INDIRECT où s'insère la référence de la terminaison

résultat: la terminaison participe passé est insérée

PROCEDURE AJOUT;

arguments : -

résultats : INDIRECT, TERMINAISON contiennent 25 terminaisons en plus

fonction : Demander le numéro du groupe à ajouter.
Calculer l'adresse où s'insérera la 1ère terminaison
Insérer les 25 terminaisons du groupe.

PROCEDURE CONSULTER;

arguments : -

résultats : Affiche une terminaison à l'écran

fonction : Saisir un numéro de groupe existant
Saisir un numéro de temps
Afficher les terminaisons correspondantes

PROCEDURE SUPPRIMER;

arguments : -

résultats : Supprime un groupe de INDIRECT et TERMINAISON

fonction : Saisir le numéro du groupe à supprimer
Supprimer les 25 terminaisons de TERMINAISON
Supprimer les 25 références correspondantes de INDIRECT
NBTER:=NBTER-somme des longueurs de toutes les terminaisons
ADR:=ADR-25

idée de solution :

- Supprimer 25 terminaisons, c'est écraser vers la gauche tous éléments de TERMINAISON à partir de l'adresse de la 1ère terminaison du groupe soit

  INDIRECT numéro du groupe - 1 X 25 + 1

- Ensuite écraser vers la gauche les 25 éléments de INDIRECT à partir de numéro du groupe - 1 X 25 + 1

- Mettre à jour toutes les références de INDIRECT à partir de numéro du groupe - 1 X 25 + 1 en les diminuant de la somme de toutes les longueurs des terminaisons supprimées.

- sauver la nouvelle version des données sur disquettes.

PROCEDURE MODIFIER;

arguments : -
résultats : remplacer une terminaison par une autre
fonction : Saisir le numéro de groupe à modifier
Saisir le numéro de temps à modifier
S'il ne s'agit pas du participe passé
Saisir le numéro de personne

Afficher l'ancienne version de cette terminaison

Lire la nouvelle version

L'enregistrer dans TERMINAISON et, si les longueurs des 2 versions sont différentes, mettre à jour les références de INDIRECT pour les éléments suivants

4. Programme CREBESCH

Même remarque que pour le programme CREFICH.
Ce programme initialise en écriture le fichier TERM, c'est à dire qu'il est écrasé et près à être créer.

5. Programme GERSEMA

a. Les commandes

Ce programme gère les fichiers FSMNCOM, FSEMEVB, FSEMISVB, FSEMSVB.

Il est possible de

- ajouter à la fin (eh oui, rien que ça!)
- consulter
- imprimer
- modifier

b. Mode d'emploi

Pour employer une de ces commandes il faut suivre les indications présentée à l'écran.

Un menu principal vous demande d'abord si vous souhaitez

- ajouter
- consulter
- imprimer
- modifier
- quitter le programme

Le programme vous demande ensuite si vous voulez traiter le fichier

- FSEMNCOM (les noms communs)
- FSEMEVB (les verbes en "E")
- FSEMISVB (les verbes en "IS")
- FSEMSVB (les verbes en "S")

si vous ajoutez

Vous pouvez ajouter tant que vous voulez dans le fichier choisi.

si vous ajoutez de l'information à propos d'un nom commun

vous donnerez successivement:

. le nombre de classes sémantiques auxquelles appartient le mot

. la liste des numéros de classes (1..3)

. les informations nécessaires à générer des phrases

. nombre d'adjectifs "avant" (0..8)

. liste des numéros d'adjectifs "avant"

. nombre d'adjectifs "après"

. liste des numéros d'adjectifs "après"

. les informations nécessaires à décoder des phrases

. nombre de classes d'adjectifs

. liste des numéros de classes

si vous ajoutez de l'information à propos d'un verbe vous donnerez successivement :

. les informations nécessaires à générer des phrases

. nombre de classes sujet
. liste des numéros de classes sujet

. nombre de classes cod
. liste des numéros de classes cod

. nombre de classes coi
. liste des numéros de classes coi

. nombre d'adverbes
. liste des numéros d'adverbe

. les informations nécessaires à décoder des phrases

. nombre de classes sujet
. liste des numéros de classes sujet

. nombre de classes cod
. liste des numéros de classes cod

. nombre de classes coi
. liste des numéros de classes coi

Après chaque ajout, leprogramme vous demande si vous avez fini

si non le processus d'ajout recommence pour le même fichier
si oui le programme revient au menu principal

si vous consultez ou imprimez

Le programme vous fait défiler le contenu du fichier choisi.

Il vous sort toutes les informations patiemment rentées ci-dessus

A la fin, le programme vous propose une autre consultation ou impression . Si vous refusez il revient au menu principal.

Si vous modifiez

Le programme vous propose :

pour une modification d'un nom commun une des modifications suivantes

sens générer

1. adjectifs avant
2. adjectifs après

sens recevoir

3. ses classes adjectifs

appartient

4. numéros des classes auxquelles il appartient

pour une modification d'un verbe une des modifications suivantes

sens recevoir

1. liste des classes sujet
2. liste des classee cod
3. liste des classes coi

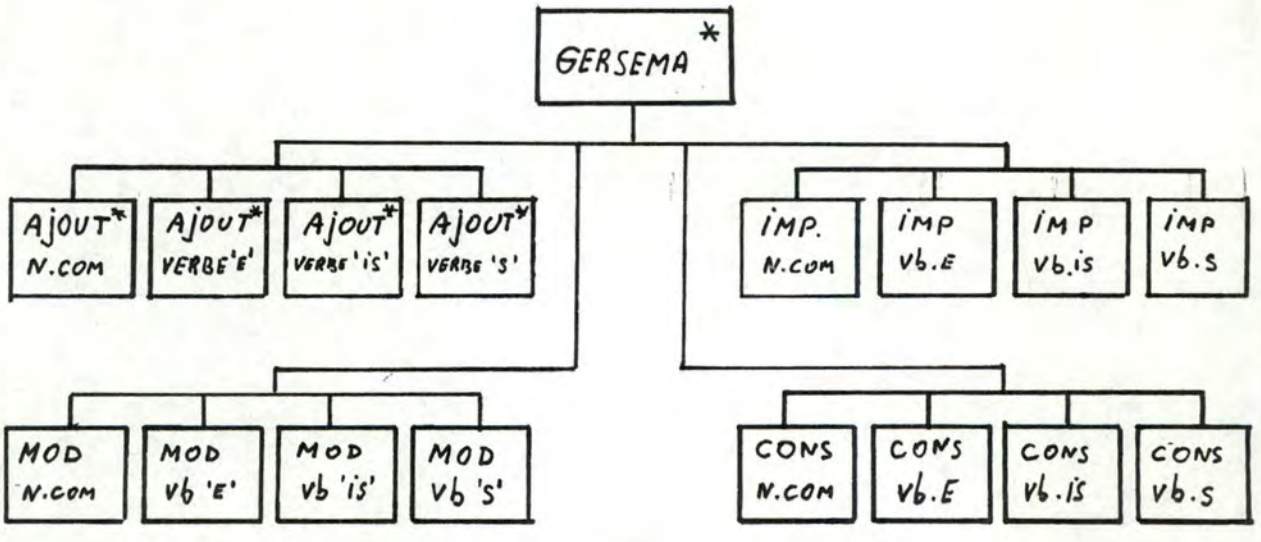
sens générer

4. listedes classes sujet
5. liste des adverbes :
6. liste des classes cod
7. liste des classes coi

à chaque modification, le programme propose l'ancienne version et demande la nouvelle version.

Après chaque modification le programme vous propose d'encore modifier le même fichier, si vous refusez le programme revient au menu principal.

c. Architecture du programme



d. Spécifications des procédures

PROCEDURE AJOUTER;

argument : -
résultat : un ou plusieurs articles en plus dans un des fichiers gérés par GERSEMA

fonction : saisir le choix du fichier à traiter et y ajouter tous les articles que veut l'utilisateur

PROCEDURE AJC;

argument : -
résultats : FSEMNC est augmenté de un ou plusieurs articles

fonction : ajouter des informations sémantiques sur les noms communs dans FSEMNC

PROCEDURES AJE, AJI, AJS;

idem AJC mais on ajoute dans les fichiers FSEMEVB, FSEMISVB ou FSEMSVB

PROCEDURE LIRESEMNON;

argument : NMOT numéro du nom commun dont on va entrer les informations sémantiques
résultat : NCSEMAN buffer dans lequel on va mémoriser les informations avant de les écrire dans le fichier
fonction : lire les informations sémantiques d'un nom commun

PROCEDURE LIRESEMVERBE;

argument : NMOT numéro du verbe dont on va entrer les informations sémantiques
résultat : VSEMAN buffer dans lequel on va mémoriser les informations avant de les écrire sur fichier
fonction : lire les informations sémantiques d'un verbe

PROCEDURE CONSULTER;

argument : -

résultat : sortie sur écran d'un des fichier

fonction : saisir le choix du fichier à consulter et sortir le contenu de ce fichier sur écran, ceci tant que l'utilisateur veut consulter des fichiers

PROCEDURES CONSC, CONSE, CONSI, CONSS

argument : -

résultat : sortie sur écran de FSEMNC, FSEMEVB, FSEMISVB ou FSEMSVB
fonction : sortir sur écran le contenu d'un des fichiers

PROCEDURE SORTIRNC;

argument : BUFF buffer qui contient le NMOTième article de FSEMNC
NMOT numéro de l'article de FSEMNC à sortir
résultat : écrire le contenu de BUFF à l'écran

PROCEDURE SORTIRVB;

argument : BUFF buffer qui contient l'article à sortir
NMOT numéro du verbe dont on doit sortir l'information sémantique

PROCEDURE IMPRIMER;

idem consulter mais les résultats sortent sur papier

se compose de

IMPC	idem	CONSC
IMPE	idem	CONSE
IMPI	idem	CONSI
IMPS	idem	CONSS
IMPRIMNC	idem	SORTIRNC
IMPRIMVB	idem	SORTIRVB

PROCEDURE MODIFIER;

argument : -

résultat : un article d'un des fichiers est modifié

fonction : saisir le choix du fichier à modifier
et modifier tous les articles de ce fichier que souhaite modifier l'utilisateur

PROCEDURE MODC;

argument : -

résultat : modifie un article du fichier FSEMNC

fonction : saisir le numéro d'un article de FSEMNC à modifier
propose un éventail de choses à modifier
modifie un élément au choix de l'utilisateur
propose d'encore modifier

PROCEDURES MODE, MODI, MODS;

idem MODC mais pour les fichiers FSEMEVB FSEMISVB ou FSEMSVB

PROCEDURE GRILLESUP;

arguments: BUFFV contient une copie de l'article à modifier
NUM numéro du verbe à modifier
résultat : BUFFV contient la copie de l'article modifié

fonction : saisir le choix de la partie d'article à modifier et la modifier

PROCEDURE NCGRILLESUP;

arguments: BUFF contient une copie de l'article à modifier
NUM numéro du nom commun à modifier
résultat : BUFF contient la copie de l'article modifié

fonction : saisir le choix de la partie d'article à modifier et la modifier

PROCEDURE NC1 NC2 NC3 NC4;

arguments: BUFF contient une copie de l'article à modifier
NUM numéro de l'article à modifier
résultat : BUFF contient une copie de l'article modifié

fonction : NC1 modifie la liste des adjectifs "avant"
NC2 ..... "après"
NC3 modifie la liste des classes adjectifs
NC4 modifie les numéros de classes auxquelles appartient le nom commun numéro NUM

PROCEDURE M1 M2 M3 M4 M5 M6 M7;

arguments :BUFFV contient une copie de l'article à modifier
NUM numéro de l'article à modifier
résultats :BUFFV contient de l'article modifié

fonction :M1 modifie la liste des classes-sujet dans le sens recevoir
M2 modifie la liste des classes cod dans le sens recevoir
M3 modifie la liste des classes coi dans le sens recevoir
M4 modifie la liste des classes sujet dans le sens générer
M5 modifie la liste des classes cod dans le sens générer
M6 modifie la liste des classes coi dans le sens générer
M7 modifie la liste des numéros d'adverbes dans le sens générer

6. Programme CRESEM

Même remarque que pour CREBESCH et CREFICH, c'est à dire qu'il initialise en écriture les fichiers FSEMNC FSEMEVB FSEMISVB et FSEMSVB. Donc nepas les exécuter, ils sont là par politesse de ma part.

7. Programme CLASSES

Ce programme gère le fichier FCLASSES.

Je rappelle qu'il s'agit du fichier des classes noms, à chaque article correspond une classe. Une classe étant une liste de numéros de noms communs.

a. Les commandes

Le programme vous permet de

- Créer le fichier (que j'ai débranché par sécurité)
- Consulter tout le fichier
- Modifier le contenu d'une classe
- Imprimer tout le fichier

b. Mode d'emploi

Le programme vous aide.

Il vous propose d'abord un menu principal qui vous permettra de choisir votre orientation :

- consulter
- modifier
- imprimer
- vous pouvez aussi créer le fichier (et donc détruire le vieux) mais il vous faudra programmer en PASCAL l'appel à la procédure TOUTCREER.

Si vous choisissez de consulter:

Le programme sort à l'écran par ordre de numéro de classe et pour chaque classe

- son numéro
- sa longueur
- la liste des numéros de noms communs qui la composent

Si vous choisissez de modifier:

Le programme vous demande le numéro de la classe à modifier et vous affiche la longueur et le contenu de cette classe.

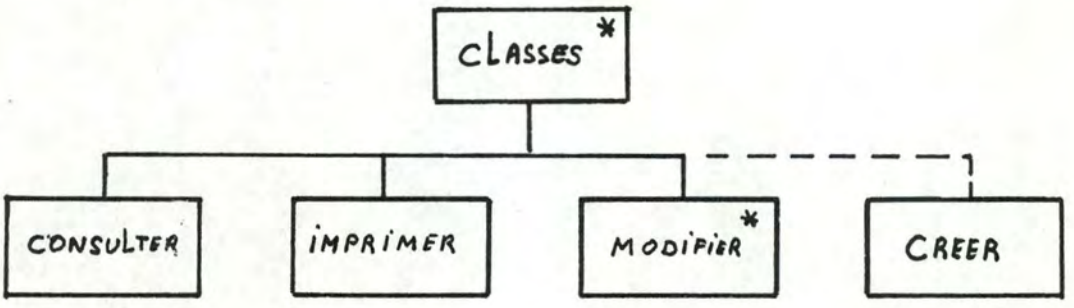
Il vous demande alors la longueur et le nouveau contenu de la classe.

Si vous choisissez d'imprimer

Le programme vous sort sur papier (si l'imprimante est branchée et de bonne humeur, il faut les deux vous verrez) par ordre de numéro de classe et pour chaque classe

- son numéro
- sa longueur
- la liste des numéros de noms communs qui la composent.

c. Architecture du programme



d. Spécifications des procédures

PROCEDURE CONSULTER;

arguments : - FCLASSES

résultats : affichage de FCLASSES à l'écran

fonction : afficher chaque article de FCLASSE à l'écran
mise en page:

. numéro de l'article
. longueur de la classe
. liste des éléments de la classe

PROCEDURE TOUTCREER ;

arguments : -

résultats : FCLASSES existe

fonction : créer le fichier et le remplir
pour chaque article, demander à l'utilisateur la longueur de la liste, les composants de la liste et enregistrer l'article

PROCEDURE MODIFIER;

argument : - FCLASSES

résultat : FCLASSES

fonction : saisir le numéro de l'article à modifier
afficher son contenu
lire son nouveau contenu et l'enregistrer

PROCEDURE IMPRIMER;

argument : - FCLASSES

résultat : impression des numéros de classes suivis des numéros des noms communs qui composent ces classes

fonction : pour chaque classe, impression de son numéro et des numéros des noms communs qui la composent

.

ANNEXE A.V

IDEE DU CONTENU DE LA BASE DE DONNEES

.LISTE DES NOMS PAR CLASSE
.LISTE DES ADJECTIFS PAR CLASSE
.LISTE DES VERBES EN 'E',EN 'IS',EN 'S'
.LISTE DES ADVERBES
.LISTE DES NOMS PROPRES

CLASSES NOMS.

Classe n°1

ami
client
élève
enfant
femme
frère
gamin
homme
instituteur
mère
père
soeur

Classe n°2

acteur
chanteur
comédien
gardien
gendarme
joueur
lutin
magicien
musicien
policier
prince
prisonnier
reine
roi
savant
troubadour
voleur

Classe n°3

boucher
boulanger
docteur
facteur
marchand
menuisier
patron
pharmacien

Classe n°4

(prénoms)

Classe n°5

chat
chien
marsupilami
oiseau
rat
souris

Classe n°6

canard
cheval
lapin
poule

Classe n°7

avion
bateau
fusée
tracteur
train
vélo
voiture

Classe n°8

école
ferme
garage
grange
maison
usine

Classe n°9

ballon
poupée
puzzle
surprise

Classe n°10

grue
machine
ordinateur
outil

Classe n°11

bière
café
eau
thé
vin

Classe n°12

biscuit
boudin
crème
dessert
fromage
gateau
jambon
pain
repas
salami
saucisse
tartine
viande

Classe n°13

bras
dent
jambe
main
oreille
pied
tête

Classe n°14

armoire
assiette
blouse
chaise
costume
couteau
cuiller
cuillère
fourchette
gilet
habit
lit
meuble
pantalon
table
tasse
téléphone
verre
vêtement

Classe n°15

bois
corde
cuivre
fer
laine
terre

Classe n°16

bêtise
chanson
conte
fable
histoire
poème
texte

Classe n°17

cinéma
cirque
concert
radio
spectacle
télévision
théâtre

Classe n°18

baffe
claque
coup

Classe n°19

idée
projet

Classe n°20

grippe
maladie
rhume
rubéole
varicelle

Classe n°21

ananas
cerise
fleur
fraise
framboise
poireau
potiron
rose
tulipe
violette

Classe n°22

calcul
grammaire
science

CLASSES ADJECTIFS.

Classe n°1

blanc
bleu
brun
gris
jaune
noir
rouge
vert

Classe n°2

bruyant
rapide

Classe n°3

beau
bon
grand
gras
petit
vieux

Classe n°4

astucieux
bête
bon
bruyant
gai
gentil
habile
malade
méchant
pauvre
triste
violent
jeune

Classe n°5

bon
bruyant
gai
gentil
triste
violent
méchant

Certains adjectifs appartiennent à plusieurs classes d'adjectifs, voyez bien la description de la base de données dans le mémoire à la page n°38

LISTE DES ADJECTIFS QUALIFICATIFS.

astucieux
beau
bête
blanc
bleu
bon
brun
bruyant
gai
gentil
grand
gris
gros
habile
jaune
jeune
malade
méchant
noir
pauvre
petit
rapide
rouge
triste
vert
vieux
violent

LISTE DES NOMS PROPRES.

Noms propres féminins	Noms propres masculins
Alice	Claude
Amélie	David
Anne	Dingo
Catherine	Fantasio
Cécile	Gaston
Christine	Joseph
Hélène	Mickey
Isabelle	Milou
Maman	Papa
Monique	Pierrot
Nathalie	René
Nicole	Serge
Sylvie	Spirou
Thérèse	Thomas
Véronique	Tintin

CORRESPONDANCE ENTRE LES NUMEROS D'ORDRE DE CONJUGAISON ET LES NUMEROS DE CONJUGAISON "BESCHERELLE"

Numéros d'ordre de conjugaison	Numéros de conjugaison Bescherelle
1	6
2	8
3	12
4	15
5	19
6	25
7	27
8	28
9	32
10	53
11	54
12	56
13	69
14	77
15	78
16	80
17	82

N.B.REFERENCE:Le nouveau Bescherelle,l'art de conjuguer
Edition Hatier

ANNEXE VI

TEXTE DES PROGRAMMES

.PROGRAMME GENERATION

.PROGRAMME CORMOT

```
(*$S++ *)

PROGRAM GENERATION;



(*  GENERE LES PHRASES DE TYPE 1, LE
    CORPS PRINCIPAL DE CE PROGRAMME
    CONSTITUE LE CORPS DE LA PROCEDURE
    GENPHTYP1 DES SPECIFICATIONS.

    LES PHRASES SORTENT A L'ECRAN

*)


USES APPLESTUFF;

(*   DECLARATION DES TYPES DE VARIABLES DU PROGRAMME  "GENERATION"    *)



CONST    NBCHAR=2000;
         NBMOT=290;
         NBMNP=17;
         NBFNP=17;
         NBVE=21;
         NBVIS=7;
         NBVS=16;
         NBNC=155;
         NBAQ=30;
         NBAM=30;
         NCH = 2000;
         NTERM = 500;



TYPE STRING4  = STRING[4];

     STRING5  = STRING[5];

     STRING8  = STRING[8];

     STRING10 = STRING[10];

     STRING20 = STRING[20];
```

```
(*SEMADJDESC = PACKED RECORD
                 NBCL:1..3;
                 LCLAS:PACKED ARRAY[1..3]OF INTEGER
               END;*)



VALCLAS=PACKED RECORD
                 LISTEMOT:ARRAY[1..20] OF INTEGER;
                 LONGLIST:INTEGER
               END;



FONC = (SUJ,COD,COIA,ADV);



SEMNCDESC = PACKED RECORD

              GENERER:PACKED RECORD
                        NBADAV:INTEGER;
                        NBADAP:INTEGER;
                        PTADAV:PACKED ARRAY[1..8]OF INTEGER;
                        PTADAP:PACKED ARRAY[1..8]OF INTEGER
                      END;

              RECEVOIR:PACKED RECORD
                         LCLASSE:PACKED ARRAY[1..5]OF BOOLEAN
                       END;

              APPARTIENT:PACKED RECORD
                           NBCL:0..3;
                           LCLAS:PACKED ARRAY[1..3]OF INTEGER
                         END

            END;
```

```
SEMVBDESC = PACKED RECORD

              CODEXISTE:BOOLEAN;

              COIAEXISTE:BOOLEAN;


              RECEVVB:PACKED RECORD

                SUJOK:PACKED RECORD
                   SNBCLAS:INTEGER;
                   STAB:PACKED ARRAY[1..25]OF BOOLEAN
                  END;

                CODOK:PACKED RECORD
                   CODNBCLAS:INTEGER;
                   CODTAB:PACKED ARRAY[1..25]OF BOOLEAN
                  END;

                COIAOK:PACKED RECORD
                   COIANBCLAS:INTEGER;
                   COIATAB:PACKED ARRAY[1..25]OF BOOLEAN
                  END

              END;


              GENERVB:PACKED RECORD

                LISTSUJ:PACKED ARRAY[1..20]OF INTEGER;
                NBSUJ:INTEGER;
                ADVTAB:PACKED ARRAY[1..10]OF INTEGER;
                NBADV:INTEGER;

                COD:PACKED RECORD
                   CODLIST:PACKED ARRAY[1..20]OF INTEGER;
                   NBCOD:INTEGER
                  END;

                COIA:PACKED RECORD
                   COIALIST:PACKED ARRAY[1..20]OF INTEGER;
                   NBCOIA:INTEGER
                  END;

              END;

            END;
```

```
(* DECLARATION DE TYPES PROPRES A LA CONJUGUAISON ------------*)


    TABTERM = PACKED ARRAY [1..NCH] OF CHAR;


    TABINDIR = PACKED ARRAY [1..NTERM] OF INTEGER;


    TABSTRING = PACKED ARRAY [1..6] OF STRING8;


    GROUPE = PACKED RECORD
                PRESENT:TABSTRING;
                IMPARFAIT:TABSTRING;
                PASSESIMPLE:TABSTRING;
                FUTURSIMPLE:TABSTRING;
                PARTPASSE:STRING4
              END;


    TERMCONJ = FILE OF GROUPE;



(* DECLARATION DES TYPES PROPRES A LA SYNTAXE ----------*)


    NATURE=(FNPROPRE,MNPROPRE,NCOM,EVERBE,ISVERBE,SVERBE,ADJQ,ADVMAN);


    SUFFIXE=PACKED RECORD
                    FSY:BOOLEAN;
                    FPY:BOOLEAN;
                    MSY:BOOLEAN;
                    MPY:BOOLEAN;
                    FS:STRING8;
                    FP:STRING8;
                    MS:STRING8;
                    MP:STRING8;
                  END;


    SUFFVB=PACKED RECORD
                     INF:STRING5;
                     CONJ:INTEGER
                   END;
```

```
DESCRIPTMOT = PACKED RECORD
                NAT:NATURE;
                NUMPARNAT:INTEGER;
                ADRESSE:INTEGER
              END;


WORD = PACKED RECORD
         ORTHO:STRING20;
         NAT:NATURE
       END;


STXNCDESC = PACKED RECORD
              CARACT:PACKED ARRAY[1..6] OF BOOLEAN;
              TERM:SUFFIXE
            END;


STXNCOM = PACKED RECORD
            STX:STXNCDESC;
            ADIND:INTEGER
          END;


STXVBDESC = PACKED RECORD
              TERMVB:SUFFVB;
              ADIND:INTEGER
            END;


STXADJDESC = PACKED RECORD
               TERMADJ:SUFFIXE;
               ADIND:INTEGER
             END;


TABI10 = PACKED ARRAY[1..10] OF INTEGER;

TABC2100 = PACKED ARRAY[1..NBCHAR] OF CHAR;

TABD290 = ARRAY[1..NBMOT] OF DESCRIPTMOT;

TABINBMNP = ARRAY[1..NBMNP] OF INTEGER;

TABINBFNP = ARRAY[1..NBFNP] OF INTEGER;

TABSNBNCOM = ARRAY[1..NBNC] OF STXNCOM;

TABSNBVB = ARRAY[1..NBVB] OF STXVBDESC;

TABSISNBV = ARRAY[1..NBVIS] OF STXVBDESC;

TABSSNBV = ARRAY[1..NBVS] OF STXVBDESC;

TABSNBADJQ = ARRAY[1..NBAQ] OF STXADJDESC;

TABINBADV = ARRAY[1..NBAM] OF INTEGER;
```

(* DECLARATION DES TYPES PROPRES A LA GENERATION *)

```
GRNOM=PACKED RECORD
        DETERMINANT:PACKED RECORD
                      D1:STRING8;
                      D2:STRING8
                    END;
        ADAV:STRING20;
        NOM:STRING20;
        ADAP:STRING20;
        INDIC:PACKED ARRAY[1..5] OF INTEGER;
      END;


SPRONOM=STRING20;


DESC=PACKED RECORD
       NC:CHAR;
       PC:CHAR;
       GC:CHAR;
       CASE TC:CHAR OF
        'P':(CP:SPRONOM);
        'G':(CG:GRNOM)
     END;


SUITCOMPOSANT=PACKED RECORD
                COMP:ARRAY[1..3] OF DESC;
                NBCOMP:INTEGER
              END;
```

```
    SUJETDESC=PACKED RECORD
                NOMBRE:CHAR;
                PERSONNE:CHAR;
                GENRE:CHAR;
                VAL:PACKED RECORD
                      CASE TYPSUJ:CHAR OF
                       'P':(PRONOM:SPRONOM);
                       'G':(GR:GRNOM);
                       'S':(ST:SUITCOMPOSANT)
                      END
              END;
```

```
(**** DECLARATION DES TYPES DE VARIABLES ***)

(*$I #5:TYPES.TEXT *)
```

```
VAR (* CE QUI EST COMMUN A TOUT LES TYPES DE MOTS *)

    LISTMOT     :TABC2100;
    INDICE      :TABD290;
    NBREMOT     :INTEGER;
    NBRECAR     :INTEGER;
    FICHPREFIXES:FILE OF WORD;

    (* CE QUI EST SPECIFIQUE AU NOMS PROPRES FEMININS *)

    TFNPROP:TABINBFNP;
    NUMFNP :INTEGER;

    (* CE QUI EST SPECIFIQUE AUX NOMS PROPRES MASCULINS *)

    TMNPROP:TABINBMNP;
    NUMMNP :INTEGER;
```

```
(* CE QUI EST SPECIFIQUE AUX NOMS COMMUNS *)

TNCOM   :TABSNBNCOM;
NUMNC   :INTEGER;
FSTXNCOM:FILE OF STXNCDESC;
BUFFNOM :SEMNCDESC;
FSEMNC  :FILE OF SEMNCDESC;


(* CE QUI EST SPECIFIQUE AU CLASSES DE NOMS *)

LGCLASSE:ARRAY[1..22] OF INTEGER;
FCLASSES:FILE OF VALCLAS;


(* CE QUI EST SPECIFIQUE AUX VERBES EN 'E','IS','S' *)

TEVERBE :TABSENBV;
NUMEV   :INTEGER;
FESTXVB :FILE OF SUFFVB;
FSEMEVB :FILE OF SEMVBDESC;

TISVERBE:TABSISNBV;
NUMISV  :INTEGER;
FISSTXVB:FILE OF SUFFVB;
FSEMISVB:FILE OF SEMVBDESC;

TSVERBE :TABSSNBV;
NUMSV   :INTEGER;
FSSTXVB :FILE OF SUFFVB;
FSEMSVB :FILE OF SEMVBDESC;

BUFFVERBE:SEMVBDESC;


(* VARIABLES PROPRE A LA CONJUGAISON *)

TERM        :TERMCONJ;
TERMINAISON:TABTERM;
INDIRECT    :TABINDIR;


(* CE QUI EST SPECIFIQUE AUX ADJECTIFS QUALIFICATIFS *)

TADJQ  :TABSNBADJQ;
NUMADJ :INTEGER;
FSTXADJ:FILE OF SUFFIXE;
(*FSEMADJ:FILE OF SEMADJDESC;*)


(* CE QUI EST SPECIFIQUE AUX ADVERBES DE MANIERES *)

TADV  :TABINBADV;
NUMADV:INTEGER;


(* VARIABLES PROPRES A LA GENERATION *)

NOVERBE,NG,TEMPS,PERS:INTEGER;
AUX,VERBCONJ          :STRING;
FONCCPL               :FONC;
COMPL                 :GRNOM;
DSUJET                :SUJETDESC;
NEXTLETTRE,NBRCOI     :CHAR;
PHRASETYP             :INTEGER;

REP :CHAR;
FINI:BOOLEAN;
```

```
    (* VARIABLE PROPRE A LA MISE EN PAGE SUR IMPRIMANTE *)

    TAILLE,LONGLIGNE:INTEGER;




PROCEDURE CHARBESCH(SOURCE:INTEGER;VAR LONGUEUR:INTEGER;
                    VAR DESTINATION:STRING);FORWARD;
PROCEDURE CHARBYTE(SOURCE:INTEGER;VAR LONGUEUR:INTEGER;
                    VAR DESTINATION:STRING20);FORWARD;
PROCEDURE CHOISIRALEAT(BI,BS:INTEGER;VAR RESULT:INTEGER);FORWARD;
PROCEDURE PREMLETTRE(MOT:STRING20;VAR LETTRE:CHAR);FORWARD;
PROCEDURE CHOISADV(VAR MOT:STRING20;LGLISTE:INTEGER;LISTE:TABI10);FORWARD;
PROCEDURE CHOISFNPROPRE(VAR MOT:STRING20);FORWARD;
PROCEDURE CHOISMNPROPRE(VAR MOT:STRING20);FORWARD;
PROCEDURE PASSAGEALALIGNE(VAR LONGLIGNE,LONGAJOUTEE:INTEGER);FORWARD;
PROCEDURE BLANC(VAR LONGLIGNE:INTEGER);FORWARD;




(********* CHARGEMENT DES DONNEES ***************)

(* SEGMENT PROCEDURE CHARGER *)

(*$I #5:CHARGER.TEXT *)



(**** CONSTRUCTION D'UN GROUPE NOM ****)

(*$I #5:GRPN.TEXT *)



(**** CONSTRUCTION D'UNE SUITE DE COMPOSANTS ****)

(*$I #5:CSUIT.TEXT *)
```

```
(********* CHARGEMENT DES DONNEES ***************)

SEGMENT PROCEDURE CHARGER;



(* ....CHARGEMENT DU VOCABULAIRE ....*)



PROCEDURE CHARGEMENT;

(* ON CHARGE LE VOCAB : 15 TONNES !!!! *)


VAR ENTREE:WORD;
    IIND,ILST:INTEGER;


 PROCEDURE CHARGFNP;
 BEGIN
   NUMFNP:=NUMFNP+1;
   INDICE[IIND].NUMPARNAT:=NUMFNP;
   TFNPROP[NUMFNP]:=IIND
 END;(*CHARGFNP*)


 PROCEDURE CHARGMNP;
 BEGIN
   NUMMNP:=NUMMNP+1;
   INDICE[IIND].NUMPARNAT:=NUMMNP;
   TMNPROP[NUMMNP]:=IIND
 END;(*CHARGMNP*)


 PROCEDURE CHARGNCOM;
 VAR NUMERO:INTEGER;
 BEGIN
   NUMNC:=NUMNC+1;
   INDICE[IIND].NUMPARNAT:=NUMNC;
   TNCOM[NUMNC].ADIND:=IIND;
   NUMERO:=NUMNC-1;
   SEEK(FSTXNCOM,NUMERO);
   GET(FSTXNCOM);
   TNCOM[NUMNC].STX:=FSTXNCOM^
 END;(*CHARGNCOM*)


 PROCEDURE CHARGEVERBE;
 VAR NUMERO:INTEGER;
 BEGIN
    NUMEV:=NUMEV+1;
    INDICE[IIND].NUMPARNAT:=NUMEV;
    TEVERBE[NUMEV].ADIND:=IIND;
    NUMERO:=NUMEV-1;
    SEEK(FESTXVB,NUMERO);
    GET(FESTXVB);
    TEVERBE[NUMEV].TERMVB:=FESTXVB^
 END;(*CHARGEVERBE*)
```

```
PROCEDURE CHARGISVERBE;
VAR NUMERO:INTEGER;
BEGIN
   NUMISV:=NUMISV+1;
   INDICE[IIND].NUMPARNAT:=NUMISV;
   TISVERBE[NUMISV].ADIND:=IIND;
   NUMERO:=NUMISV-1;
   SEEK(FISSTXVB,NUMERO);
   GET(FISSTXVB);
   TISVERBE[NUMISV].TERMVB:=FISSTXVB^
END;(*CHARGISVERBE*)


PROCEDURE CHARGSVERBE;
VAR NUMERO:INTEGER;
BEGIN
   NUMSV:=NUMSV+1;
   INDICE[IIND].NUMPARNAT:=NUMSV;
   TSVERBE[NUMSV].ADIND:=IIND;
   NUMERO:=NUMSV-1;
   SEEK(FSSTXVB,NUMERO);
   GET(FSSTXVB);
   TSVERBE[NUMSV].TERMVB:=FSSTXVB^
END;(*CHARGSVERBE*)


PROCEDURE CHARGAQ;
VAR NUMERO:INTEGER;
BEGIN
  NUMADJ:=NUMADJ+1;
  INDICE[IIND].NUMPARNAT:=NUMADJ;
  TADJQ[NUMADJ].ADIND:=IIND;
  NUMERO:=NUMADJ-1;
  SEEK(FSTXADJ,NUMERO);
  GET(FSTXADJ);
  TADJQ[NUMADJ].TERMADJ:=FSTXADJ^
END;(*CHARGAQ*)


PROCEDURE CHARGAM;
BEGIN
  NUMADV:=NUMADV+1;
  INDICE[IIND].NUMPARNAT:=NUMADV;
  TADV[NUMADV]:=IIND
END;(*CHARGAM*)
```

```
BEGIN
  NUMMNP:=0;
  NUMFNP:=0;
  NUMNC:=0;
  NUMEV:=0;
  NUMISV:=0;
  NUMSV:=0;
  NUMADJ:=0;
  NUMADV:=0;
  RESET(FSTXNCOM,'#5:STXNCOM.DATA');
  RESET(FESTXVB,'#5:STXEVERBE.DATA');
  RESET(FISSTXVB,'#5:STXISVERBE.DATA');
  RESET(FSSTXVB,'#5:STXSVERBE.DATA');
  RESET(FSTXADJ,'#5:STXADJ.DATA');
  RESET(FICHPREFIXES,'#5:TEST2.DATA');
  IIND:=0;
  ILST:=1;
  LISTMOT[ILST]:='@';
  WHILE NOT EOF(FICHPREFIXES) DO
     BEGIN
       IIND:=IIND+1;
       ILST:=ILST+1;
       ENTREE:=FICHPREFIXES^;
       MOVELEFT(ENTREE.ORTHO[1],LISTMOT[ILST],LENGTH(ENTREE.ORTHO));
       INDICE[IIND].ADRESSE:=ILST;
       INDICE[IIND].NAT:=ENTREE.NAT;
       CASE ENTREE.NAT OF
            FNPROPRE:CHARGFNP;
            MNPROPRE:CHARGMNP;
            NCOM    :CHARGNCOM;
            EVERBE  :CHARGEVERBE;
            ISVERBE :CHARGISVERBE;
            SVERBE  :CHARGSVERBE;
            ADJQ    :CHARGAQ;
            ADVMAN  :CHARGAM;
           END;
       ILST:=ILST+LENGTH(ENTREE.ORTHO);
       LISTMOT[ILST]:='@';
       GET(FICHPREFIXES)
     END;
  INDICE[IIND+1].ADRESSE:=ILST+1;
  LISTMOT[ILST+1]:='Z';
  LISTMOT[ILST+2]:='Z';
  LISTMOT[ILST+3]:='@';
  NBRECAR:=ILST+3;
  NBREMOT:=IIND+1;
  CLOSE(FSTXNCOM,LOCK);
  CLOSE(FESTXVB,LOCK);
  CLOSE(FISSTXVB,LOCK);
  CLOSE(FSSTXVB,LOCK);
  CLOSE(FSTXADJ,LOCK);
  CLOSE(FICHPREFIXES,LOCK)
END;(*CHARGEMENT*)
```

```
(*..... CHARGEMENT DU BESCHERELLE ....*)

PROCEDURE CHARGTERM;

VAR K    :INTEGER;   (* NO DE PERSONNE   1:1ERE SING  4:1ERE PLUR
                                          2:2EME SING  5:2EME PLUR
                                          3:3EME SING  6:3EME PLUR *)
    LG   :INTEGER;   (* LONGUEUR D'UNE TERMINAISON                 *)
    INTER:STRING;    (* INTERMEDIAIRE QUI PEUT CONTENIR UN CHAMPS
                        D'UN ARTICLE DU FICHIER TERM               *)
    ADR  :INTEGER;   (* PREMIERE ADRESSE LIBRE DANS INDIRECT       *)
    NBTER:INTEGER;   (* PREMIERE ADRESSE LIBRE DANS TERMINAISON    *)
    GRP  :GROUPE;    (* INTERMEDIAIRE QUI PEUT CONTENIR UN
                        ARTICLE DU FICHIER TERM                    *)


BEGIN
  ADR:=1;
  NBTER:=2;
  TERMINAISON[1]:='@';
  RESET(TERM,'#5:BESCH.DATA');
  WHILE NOT EOF(TERM) DO
    BEGIN
      GRP:=TERM^;

      FOR K:=1 TO 6 DO
        BEGIN
          LG:=LENGTH(GRP.PRESENT[K]);
          INTER:=GRP.PRESENT[K];
          MOVELEFT(INTER[1],TERMINAISON[NBTER],LG);
          INDIRECT[ADR]:=NBTER;
          NBTER:=NBTER+LG;
          ADR:=ADR+1
        END;

      FOR K:=1 TO 6 DO
        BEGIN
          LG:=LENGTH(GRP.IMPARFAIT[K]);
          INTER:=GRP.IMPARFAIT[K];
          MOVELEFT(INTER[1],TERMINAISON[NBTER],LG);
          INDIRECT[ADR]:=NBTER;
          NBTER:=NBTER+LG;
          ADR:=ADR+1
        END;

      FOR K:=1 TO 6 DO
        BEGIN
          LG:=LENGTH(GRP.PASSESIMPLE[K]);
          INTER:=GRP.PASSESIMPLE[K];
          MOVELEFT(INTER[1],TERMINAISON[NBTER],LG);
          INDIRECT[ADR]:=NBTER;
          NBTER:=NBTER+LG;
          ADR:=ADR+1
        END;

      FOR K:=1 TO 6 DO
        BEGIN
          LG:=LENGTH(GRP.FUTURSIMPLE[K]);
          INTER:=GRP.FUTURSIMPLE[K];
          MOVELEFT(INTER[1],TERMINAISON[NBTER],LG);
          INDIRECT[ADR]:=NBTER;
          NBTER:=NBTER+LG;
          ADR:=ADR+1
        END;

      LG:=LENGTH(GRP.PARTPASSE);
      MOVELEFT(GRP.PARTPASSE[1],TERMINAISON[NBTER],LG);
      INDIRECT[ADR]:=NBTER;
      NBTER:=NBTER+LG;
      ADR:=ADR+1;
      GET(TERM)
    END; (* END WHILE NOT EOF(TERM) *)
  CLOSE(TERM,LOCK);
  INDIRECT[ADR]:=NBTER;
END;(*CHARGTERM*)



(* ....CHARGEMENT DE LA LONGUEUR DES CLASSES....*)


PROCEDURE CHARGLONGUEUR;

VAR I:INTEGER;

BEGIN
  I:=1;
  RESET(FCLASSES,'#5:CLASSES.DATA');
  WHILE NOT EOF(FCLASSES) DO
    BEGIN
      LGCLASSE[I]:=FCLASSES^.LONGLIST;
      GET(FCLASSES);
      I:=I+1
    END;
  LGCLASSE[4]:=NUMMNP+NUMFNP;
  (* LA CLASSE 4 EST LA CLASSE DES NOMS PROPRES.POURQUOI LA 4 ?
     PARCE QUE JE M'EN SUIS RENDU COMPTE TROP TARD POUR CHANGER. *)
  CLOSE(FCLASSES,LOCK)
END;(*CHARGLONGUEUR*)



BEGIN
  GOTOXY(0,6);
  WRITELN('         UNE PETITE MINUTE DE SILENCE     ');
  CHARGEMENT;
  CHARGTERM;
  CHARGLONGUEUR
END;


(******** FIN DU CHARGEMENT DES DONNEES ***********)
```

```
SEGMENT PROCEDURE CONSGRPNOM(NOMBRE,GENRE:CHAR;TYPEVB,INDVERBE:INTEGER;
                             VAR FONCTION:FONC;VAR GRN:GRNOM);



TYPE TABI5=ARRAY[1..5] OF INTEGER;
     TABB6=PACKED ARRAY[1..6] OF BOOLEAN;

VAR CARACT:TABB6;




PROCEDURE DETCHEMGRPNOM(NOMBRE,GENRE:CHAR;VAR CHEMIN:TABI5);

VAR I,RES,RES2:INTEGER;


  PROCEDURE DETCHEMQUAL(VAR CHEMIN:TABI5);

  VAR RES:INTEGER;
  BEGIN
     (*WRITE('NOM SEUL/ADJ.QUAL+NOM/NOM+ADJ.QUAL ?(1,2,3): ');
     READLN(RES);*)

     CHEMIN[4]:=1;
     CHOISIRALEAT(1,6,RES);
     CASE RES OF
      2,3:CHEMIN[3]:=1;
      4  :CHEMIN[5]:=1
      END
  END;(*DETCHEMQUAL*)
```

```
BEGIN
   FOR I:=1 TO 5 DO CHEMIN[I]:=0;
   IF NOMBRE='S'
      THEN
         BEGIN
            (*WRITELN('[1] ADJ DEF');
            WRITELN('[2] ADJ IND');
            WRITELN('[3] ADJ POS');
            WRITELN('[4] ADJ DEM');
            WRITE  ('-------> ');
            READLN(RES);*)

            CHOISIRALEAT(1,4,RES);
            CHEMIN[1]:=RES;
         END
      ELSE
         BEGIN
            (*WRITELN('[1] ADJ DEF');
            WRITELN('[2] ADJ IND');
            WRITELN('[3] ADJ POS');
            WRITELN('[4] ADJ DEM');
            WRITELN('[5] ADJ NUM');
            WRITE('-------> ');
            READLN(RES);*)

            CHOISIRALEAT(1,5,RES);
            CASE RES OF
             1,3,4:BEGIN
                      (*WRITE('SANS OU AVEC ADJ NUM ?(0/1): ');
                      READLN(RES2);*)
                      CHOISIRALEAT(1,2,RES2);

                      CHEMIN[1]:=RES;
                      CHEMIN[2]:=RES2-1
                   END;
                2:CHEMIN[1]:=RES;
                5:CHEMIN[2]:=1
            END
         END;
   DETCHEMQUAL(CHEMIN)
END;(*DETCHEMGRPNOM*)
```

```
PROCEDURE GARNIR(NOMBRE,GENRE:CHAR;CHEMIN:TABI5;VAR CARACT:TABB6);
VAR I:INTEGER;
BEGIN
   FOR I:=1 TO 6 DO
      CARACT[I]:=FALSE;
   IF GENRE='F'
       THEN IF NOMBRE='S'
                THEN CARACT[1]:=TRUE
                ELSE CARACT[2]:=TRUE
       ELSE IF NOMBRE='S'
                THEN CARACT[3]:=TRUE
                ELSE CARACT[4]:=TRUE;
   IF CHEMIN[3]<>0 THEN CARACT[5]:=TRUE;
   IF CHEMIN[5]<>0 THEN CARACT[6]:=TRUE
END;(*GARNIR*)
```

```
PROCEDURE CONSTRUIREGRPNOM(NOMBRE,GENRE:CHAR;VAR FONCTION:FONC;CARACT:TABB6;
                           TYPEVB,INDVERBE:INTEGER;VAR GRN:GRNOM);


VAR I,IDXMOT:INTEGER;
    MOTQUISUIT:STRING20;


(* PROCEDURES INTERNES A "CONSTRUIREGRPNOM" *)



PROCEDURE ADDEFGEN(NOMBRE,GENRE:CHAR;MOTQUISUIT:STRING20;VAR RESULTAT:STRING8);
VAR LETTRE:CHAR;
BEGIN
   IF NOMBRE='S'
      THEN
        BEGIN
           PREMLETTRE(MOTQUISUIT,LETTRE);
           IF GENRE='F' THEN RESULTAT:='LA'
                        ELSE RESULTAT:='LE';
           CASE LETTRE OF
            'A','E','I','O','U','Y','H':RESULTAT:='L'''
            END
        END
      ELSE RESULTAT:='LES'
END;(*ADDEFGEN*)


PROCEDURE ADINFGEN(NOMBRE,GENRE:CHAR;VAR RESULTAT:STRING8);
BEGIN
   IF NOMBRE='S'
      THEN IF GENRE='F'
                THEN RESULTAT:='UNE'
                ELSE RESULTAT:='UN'
      ELSE RESULTAT:='DES'
END;(*ADINFGEN*)
```

```
PROCEDURE ADPOSGEN(NOMBRE,GENRE:CHAR;MOTQUISUIT:STRING20;VAR RESULTAT:STRING8);
VAR NBPOSS,PERS:INTEGER;
   LETTRE:CHAR;
BEGIN
   CHOISIRALEAT(1,3,PERS);
   IF NOMBRE='S'
      THEN
        BEGIN
           CHOISIRALEAT(1,2,NBPOSS);
           IF GENRE='M'
              THEN
                CASE NBPOSS OF
                 1:CASE PERS OF
                    1:RESULTAT:='MON';
                    2:RESULTAT:='TON';
                    3:RESULTAT:='SON'
                   END;
                 2:CASE PERS OF
                    1:RESULTAT:='NOTRE';
                    2:RESULTAT:='VOTRE';
                    3:RESULTAT:='LEUR'
                   END
                END
              ELSE
                CASE NBPOSS OF
                 1:CASE PERS OF
                    1:BEGIN
                        PREMLETTRE(MOTQUISUIT,LETTRE);
                        RESULTAT:='MA';
                        CASE LETTRE OF
                        'A','E','I','O','U','Y','H':RESULTAT:='MON'
                        END
                      END;
                    2:BEGIN
                        PREMLETTRE(MOTQUISUIT,LETTRE);
                        RESULTAT:='TA';
                        CASE LETTRE OF
                        'A','E','I','O','U','Y','H':RESULTAT:='TON'
                        END
                      END;
                    3:BEGIN
                        PREMLETTRE(MOTQUISUIT,LETTRE);
                        RESULTAT:='SA';
                        CASE LETTRE OF
                        'A','E','I','O','U','Y','H':RESULTAT:='SON'
                        END
                      END;
                   END;
                 2:CASE PERS OF
                    1:RESULTAT:='NOTRE';
                    2:RESULTAT:='VOTRE';
                    3:RESULTAT:='LEUR'
                   END
                END
        END
      ELSE
        CASE PERS OF
        1:RESULTAT:='NOS';
        2:RESULTAT:='VOS';
        3:RESULTAT:='LEURS'
        END
END;(*ADPOSGEN*)
```

```
PROCEDURE ADDEMGEN(NOMBRE,GENRE:CHAR;MOTQUISUIT:STRING20;VAR RESULTAT:STRING8);
VAR LETTRE:CHAR;
BEGIN
   IF NOMBRE='P'
       THEN RESULTAT:='CES'
       ELSE IF GENRE='F'
                THEN RESULTAT:='CETTE'
                ELSE BEGIN
                       PREMLETTRE(MOTQUISUIT,LETTRE);
                       RESULTAT:='CE';
                       CASE LETTRE OF
                        'A','E','I','O','U','Y','H':RESULTAT:='CET'
                        END
                     END
END;(*ADDEMGEN*)


PROCEDURE ADNUMGEN(VAR RESULTAT:STRING8);
VAR RES:INTEGER;
BEGIN
   CHOISIRALEAT(2,10,RES);
   CASE RES OF
    2:RESULTAT:='DEUX';
    3:RESULTAT:='TROIS';
    4:RESULTAT:='QUATRE';
    5:RESULTAT:='CINQ';
    6:RESULTAT:='SIX';
    7:RESULTAT:='SEPT';
    8:RESULTAT:='HUIT';
    9:RESULTAT:='NEUF';
   10:RESULTAT:='DIX'
  END
END;(*ADNUMGEN*)


PROCEDURE CHERCHENOM(VAR FONCTION:FONC;CARACT:TABB6;TYPEVB,INDVERBE:INTEGER;
                     NOMBRE,GENRE:CHAR;VAR NOM:STRING20;VAR IDXCHOIX:INTEGER);

TYPE TABI20=ARRAY[1..20] OF INTEGER;
     TABI10=ARRAY[1..10] OF INTEGER;


VAR BUFFVERBE:SEMVBDESC;
    LISTE10:TABI10;
    LISTE:TABI20;
    LGLISTE:INTEGER;


PROCEDURE CHOISIRMOT(VAR LISTCLASSE:TABI20;VAR NBCLASSE:INTEGER;CARACT:TABB6;
                     VAR IDXCHOIX:INTEGER);

  TYPE TABI3  = ARRAY[1..3] OF INTEGER;
       TABI50 = ARRAY [1..50] OF INTEGER;



  VAR LGTAB,I,TOTAL,NO,RES:INTEGER;
      TAB50:TABI50;
```

```
PROCEDURE TOTALMOT (LISTCLASSE:TABI20;NBCLASSE:INTEGER;VAR TOTAL:INTEGER);

VAR I:INTEGER;

BEGIN
  TOTAL:=0;
  FOR I:=1 TO NBCLASSE DO
    IF LISTCLASSE[I] <> 4
      THEN TOTAL:=TOTAL + LGCLASSE[LISTCLASSE[I]]
END;(*TOTALMOT*)


PROCEDURE CHOISICLASSE(LISTE:TABI20;TOTAL:INTEGER;VAR NO:INTEGER);
VAR VAL,RES:INTEGER;
BEGIN
  NO:=0;
  VAL:=0;
  CHOISIRALEAT(1,TOTAL,RES);
  WHILE VAL < RES DO
    BEGIN
      NO:=NO+1;
      IF LISTE[NO] <> 4 THEN VAL:=LGCLASSE[LISTE[NO]]+VAL;
    END;
END;(*CHOISICLASSE*)


PROCEDURE COMPACTE(VAR LISTE:TABI20;VAR LGLISTE:INTEGER;NUMERO:INTEGER);
VAR I:INTEGER;
BEGIN
  FOR I:= NUMERO TO LGLISTE-1 DO
    LISTE[I]:=LISTE[I+1];
  LGLISTE:=LGLISTE-1
END;(*COMPACTE*)


PROCEDURE REMPLTAB  (NUM:INTEGER;CARACT:TABB6;
                     VAR TAB50:TABI50;VAR LGTAB:INTEGER);

VAR I:INTEGER;
    VALABLE:BOOLEAN;
    CAROBS:TABB6;


  PROCEDURE COMPARE(CARDEM,CAROBS:TABB6;VAR VALABLE:BOOLEAN);
  VAR I:INTEGER;
  BEGIN
    VALABLE:=TRUE;
    I:=1;
    WHILE (I<=6) AND (VALABLE) DO
       IF CARDEM[I]=CAROBS[I] THEN I:=I+1
                              ELSE IF CAROBS[I]=TRUE
                                      THEN I:=I+1
                                      ELSE VALABLE:=FALSE
  END;(*COMPARE*)
```

```
BEGIN
   SEEK(FCLASSES,NUM-1);
   GET(FCLASSES);
   FOR I := 1 TO FCLASSES^.LONGLIST DO
     BEGIN
       CAROBS:=TNCOM[FCLASSES^.LISTEMOT[I]].STX.CARACT;
       COMPARE(CARACT,CAROBS,VALABLE);
       IF VALABLE THEN
                    BEGIN
                      LGTAB:=LGTAB+1;
                      TAB50[LGTAB]:=FCLASSES^.LISTEMOT[I]
                    END
     END
END;(*REMPLTAB*)
```

```
(* DEBUT DE "CHOISIRMOT" *)

BEGIN
  LGTAB:=0;
  I:=1;
  RESET(FCLASSES,'#5:CLASSES.DATA');
  TOTALMOT(LISTCLASSE,NBCLASSE,TOTAL);
  WHILE (I<=3) AND (TOTAL > 0) DO
       BEGIN
          CHOISICLASSE(LISTCLASSE,TOTAL,NO);
          TOTAL:=TOTAL-LGCLASSE[LISTCLASSE[NO]];
          REMPLTAB(LISTCLASSE[NO],CARACT,TAB50,LGTAB);
          COMPACTE(LISTCLASSE,NBCLASSE,NO);
          I:=I+1
       END;
  CLOSE(FCLASSES,LOCK);
  IF LGTAB > 0 THEN
                 BEGIN
                   CHOISIRALEAT(1,LGTAB,RES);
                   IDXCHOIX:=TAB50[RES]
                 END
               ELSE IDXCHOIX:=0
END;(*CHOISIRMOT*)
```

```
PROCEDURE ACCORDMOT(INDMOTCHOISI:INTEGER;NOMBRE,GENRE:CHAR;VAR MOT:STRING20);
VAR LONGMOT,ADR,IND:INTEGER;
    TERM:STRING8;

BEGIN
   IND:=TNCOM[INDMOTCHOISI].ADIND;
   ADR:=INDICE[IND].ADRESSE;
   LONGMOT:=INDICE[IND+1].ADRESSE-ADR-1;
   CHARBYTE(ADR,LONGMOT,MOT);
   IF GENRE='F' THEN IF NOMBRE='S'
                          THEN TERM:=TNCOM[INDMOTCHOISI].STX.TERM.FS
                          ELSE TERM:=TNCOM[INDMOTCHOISI].STX.TERM.FP
                ELSE IF NOMBRE='S'
                          THEN TERM:=TNCOM[INDMOTCHOISI].STX.TERM.MS
                          ELSE TERM:=TNCOM[INDMOTCHOISI].STX.TERM.MP;
   MOT:=CONCAT(MOT,TERM)
END;(*ACCORDMOT*)
```

```
PROCEDURE TRTERREUR (VAR FONCTION:FONC;VAR LIST:TAB120;VAR LGLSTE:INTEGER);

BEGIN
  CASE FONCTION OF

    COD:IF BUFFVERBE.COIAEXISTE
        THEN
          BEGIN
            LIST:=BUFFVERBE.GENERVB.COIA.COIALIST;
            LGLSTE:=BUFFVERBE.GENERVB.COIA.NBCOIA;
            FONCTION:=COIA
          END
        ELSE FONCTION:=ADV;
   COIA:IF BUFFVERBE.CODEXISTE
        THEN
          BEGIN
            LIST:=BUFFVERBE.GENERVB.COD.CODLIST;
            LGLSTE:=BUFFVERBE.GENERVB.COD.NBCOD;
            FONCTION:=COD
          END
        ELSE FONCTION:=ADV
  END
END;(*TRTERREUR*)
```

```
(* DEBUT CHERCHENOM *)

BEGIN

  CASE TYPEVB OF

   1:BEGIN
       RESET(FSEMEVB,'#5:SEMEVB.DATA');
       SEEK(FSEMEVB,INDVERBE-1);
       GET(FSEMEVB);
       BUFFVERBE:=FSEMEVB^;
       CLOSE(FSEMEVB,LOCK);
     END;
   2:BEGIN
       RESET(FSEMISVB,'#5:SEMISVB.DATA');
       SEEK(FSEMISVB,INDVERBE-1);
       GET(FSEMISVB);
       BUFFVERBE:=FSEMISVB^;
       CLOSE(FSEMISVB,LOCK);
     END;
   3:BEGIN
       RESET(FSEMSVB,'#5:SEMSVB.DATA');
       SEEK(FSEMSVB,INDVERBE-1);
       GET(FSEMSVB);
       BUFFVERBE:=FSEMSVB^;
       CLOSE(FSEMSVB,LOCK)
     END
  END;


  CASE FONCTION OF

   SUJ:BEGIN
         LISTE:=BUFFVERBE.GENERVB.LISTSUJ;
         LGLISTE:=BUFFVERBE.GENERVB.NBSUJ;
       END;

   COD:IF BUFFVERBE.CODEXISTE
         THEN
           BEGIN
             LISTE:=BUFFVERBE.GENERVB.COD.CODLIST;
             LGLISTE:=BUFFVERBE.GENERVB.COD.NBCOD;
           END
         ELSE TRTERREUR(FONCTION,LISTE,LGLISTE);

   COIA:IF BUFFVERBE.COIAEXISTE
         THEN
           BEGIN
             LISTE:=BUFFVERBE.GENERVB.COIA.COIALIST;
             LGLISTE:=BUFFVERBE.GENERVB.COIA.NBCOIA;
           END
         ELSE TRTERREUR(FONCTION,LISTE,LGLISTE)
  END;

  IF FONCTION <> ADV (* CAR TRTERREUR PEUT ETRE ORIENTE VERS CETTE OPTION *)

     THEN
       BEGIN
         CHOISIRMOT(LISTE,LGLISTE,CARACT,IDXCHOIX);
         IF IDXCHOIX <> 0
            THEN ACCORDMOT(IDXCHOIX,NOMBRE,GENRE,NOM)
            ELSE FONCTION:=ADV (* NOUVELLE MODIFICATION *)
       END;

  IF FONCTION=ADV

     THEN
       BEGIN
         LISTE1D:=BUFFVERBE.GENERVB.ADVTAB;
         LGLISTE:=BUFFVERBE.GENERVB.NBADV;
         CHOISADV(NOM,LGLISTE,LISTE1D);
       END
END;(*CHERCHENOM*)



PROCEDURE ACCORDQUAL(ADRINDICE:INTEGER;NOMBRE,GENRE:CHAR;VAR ADJ:STRING20);
VAR LONGMOT,ADR,IND:INTEGER;
    TERM:STRING[8];

  BEGIN
     IND:=TADJQ[ADRINDICE].ADIND;
     ADR:=INDICE[IND].ADRESSE;
     LONGMOT:=INDICE[IND+1].ADRESSE-ADR-1;
     CHARBYTE(ADR,LONGMOT,ADJ);
     IF GENRE='F' THEN IF NOMBRE='S'
                          THEN TERM:=TADJQ[ADRINDICE].TERMADJ.FS
                          ELSE TERM:=TADJQ[ADRINDICE].TERMADJ.FP
                  ELSE IF NOMBRE='S'
                          THEN TERM:=TADJQ[ADRINDICE].TERMADJ.MS
                          ELSE TERM:=TADJQ[ADRINDICE].TERMADJ.MP;
     ADJ:=CONCAT(ADJ,TERM)
  END;(*ACCORDQUAL*)


PROCEDURE GENAVQUAL(NUMMOT:INTEGER;NOMBRE,GENRE:CHAR;VAR ADJ:STRING20);
VAR NOADJ,NBADJ,RES:INTEGER;
BEGIN
   RESET(FSEMNC,'#5:SEMNC.DATA');
   SEEK(FSEMNC,NUMMOT-1);
   GET(FSEMNC);
   NBADJ:=FSEMNC^.GENERER.NBADAV;
   IF NBADJ=0 THEN WRITE('ERREUR:N.COMM NO ',NUMMOT,' N''A PAS D''A.Q AVANT')
              ELSE CHOISIRALEAT(1,NBADJ,RES);
   NOADJ:=FSEMNC^.GENERER.PTADAV[RES];
   ACCORDQUAL(NOADJ,NOMBRE,GENRE,ADJ);
   CLOSE(FSEMNC,LOCK)
END;(*GENAVQUAL*)



PROCEDURE GENAPQUAL(NUMMOT:INTEGER;NOMBRE,GENRE:CHAR;VAR ADJ:STRING20);
VAR NOADJ,NBADJ,RES:INTEGER;
BEGIN
   RESET(FSEMNC,'#5:SEMNC.DATA');
   SEEK(FSEMNC,NUMMOT-1);
   GET(FSEMNC);
   NBADJ:=FSEMNC^.GENERER.NBADAP;
   IF NBADJ=0 THEN WRITE('ERREUR:N.COMM NO ',NUMMOT,' N''A PAS D''A.Q APRES')
              ELSE CHOISIRALEAT(1,NBADJ,RES);
   NOADJ:=FSEMNC^.GENERER.PTADAP[RES];
   ACCORDQUAL(NOADJ,NOMBRE,GENRE,ADJ);
   CLOSE(FSEMNC,LOCK)
END;(*GENAPQUAL*)
```

```
(* "CONSTRUIREGRPNOM"...PRINCIPAL *)


BEGIN
  CHERCHENOM(FONCTION,CARACT,TYPEVB,INDVERBE,NOMBRE,GENRE,GRN.NOM,IDXMOT);
  IF FONCTION = ADV
     THEN
       BEGIN
         FOR I:=1 TO 5 DO GRN.INDIC[I]:=0;
         GRN.INDIC[4]:=1
       END
     ELSE
       BEGIN
         FOR I:=5 DOWNTO 1 DO
           BEGIN
             CASE I OF
               1:BEGIN
                   IF (GRN.INDIC[2]<>0)
                         THEN MOTQUISUIT:=GRN.DETERMINANT.D2
                         ELSE IF (GRN.INDIC[3]<>0)
                                 THEN MOTQUISUIT:=GRN.ADAV
                                 ELSE MOTQUISUIT:=GRN.NOM;
                   CASE GRN.INDIC[I] OF
                        1:ADDEFGEN(NOMBRE,GENRE,MOTQUISUIT,GRN.DETERMINANT.D1);
                        2:ADINFGEN(NOMBRE,GENRE,GRN.DETERMINANT.D1);
                        3:ADPOSGEN(NOMBRE,GENRE,MOTQUISUIT,GRN.DETERMINANT.D1);
                        4:ADDEMGEN(NOMBRE,GENRE,MOTQUISUIT,GRN.DETERMINANT.D1);
                      END;
                 END;
               2:IF GRN.INDIC[I]=1 THEN ADNUMGEN(GRN.DETERMINANT.D2);
               3:IF GRN.INDIC[I]=1 THEN GENAVQUAL(IDXMOT,NOMBRE,GENRE,GRN.ADAV);
               4:;
               5:IF GRN.INDIC[I]=1 THEN GENAPQUAL(IDXMOT,NOMBRE,GENRE,GRN.ADAP)
              END
           END
       END
END;(*CONSTRUIREGRPNOM*)




(* CONSGRPNOM...PRINCIPAL *)
BEGIN
   DETCHEMGRPNOM(NOMBRE,GENRE,GRN.INDIC);
   GARNIR(NOMBRE,GENRE,GRN.INDIC,CARACT);
   CONSTRUIREGRPNOM(NOMBRE,GENRE,FONCTION,CARACT,TYPEVB,INDVERBE,GRN)
END;(*CONSGRPNOM*)



(* OUF! ,  MAIS C'ETAIT GAI QUAND MEME, C'ETAIT UNE DES PLUS BELLE *)
```

```
SEGMENT PROCEDURE CONSSUITE(PERSONNE,GENRE:CHAR;TYPEVB,INDVERBE:INTEGER;
                            VAR SUITE:SUITCOMPOSANT);


VAR COMPOSANT:DESC;

(* PROCEDURES INTERNES A "CONSSUITE" *)



SEGMENT PROCEDURE DETSUITE(PERSONNE,GENRE:CHAR;VAR SUITE:SUITCOMPOSANT);



 PROCEDURE DETNBCOMP(VAR NBCOMP:INTEGER);
 BEGIN
   (*WRITE('NOMBRE DE COMPOSANTS ?(2/3) : ');
   READLN(NBCOMP)*)
   CHOISIRALEAT(2,3,NBCOMP)
 END;(*DETNBCOMP*)



 PROCEDURE DET2SUITE(PERSONNE,GENRE:CHAR;VAR SUITE:SUITCOMPOSANT);

 VAR RES,I:INTEGER;

 BEGIN
    CASE PERSONNE OF
     '1','2':BEGIN
                (*WRITE('COMP1:2 OU 3EME PERS ?(2/3) : ');
                READLN(RES);*)
                CHOISIRALEAT(2,3,RES);
                IF RES=2 THEN SUITE.COMP[1].PC:='2'
                         ELSE SUITE.COMP[1].PC:='3';

                (*WRITE('COMP1:SING OU PLUR ?(S/P)    : ');
                READLN(RES);*)
                CHOISIRALEAT(1,2,RES);
                IF RES=1 THEN SUITE.COMP[1].NC:='S'
                         ELSE SUITE.COMP[1].NC:='P';

                CASE SUITE.COMP[1].PC OF
                 '3':BEGIN
                        (*WRITE('COMP1:PRON OU GRPNOM?(P/G) : ');
                        READLN(RES);*)
                        CHOISIRALEAT(1,2,RES);
                        IF RES=1 THEN SUITE.COMP[1].TC:='P'
                                 ELSE SUITE.COMP[1].TC:='G';
                     END;
                 '2':SUITE.COMP[1].TC:='P'
                 END;
                (*WRITE('COMP2:SING OU PLUR ?(S/P) : ');
                READLN(RES);*)
                CHOISIRALEAT(1,2,RES);
                IF RES=1 THEN SUITE.COMP[2].NC:='S'
                         ELSE SUITE.COMP[2].NC:='P';

                SUITE.COMP[2].PC:=PERSONNE;
                SUITE.COMP[2].TC:='P'
              END;
```

```
        '3':FOR I:=1 TO 2 DO
              BEGIN
                 (*WRITE('COMP',I,':SING OU PLUR ?(S/P) : ');
                 READLN(RES);*)
                 CHOISIRALEAT(1,2,RES);
                 IF RES=1 THEN SUITE.COMP[I].NC:='S'
                          ELSE SUITE.COMP[I].NC:='P';

                 (*WRITE('COMP',I,':PRON OU GRPNOM?(P/G): ');
                 READLN(RES);*)
                 CHOISIRALEAT(1,2,RES);
                 IF RES=1 THEN SUITE.COMP[I].TC:='P'
                          ELSE SUITE.COMP[I].TC:='G';

                 SUITE.COMP[I].PC:=PERSONNE
              END;
      END;
     IF GENRE='F' THEN
                   BEGIN
                     SUITE.COMP[1].GC:='F';
                     SUITE.COMP[2].GC:='F'
                   END
                 ELSE
                   BEGIN
                     (*WRITE('COMP1:FEM OU MASC ?(F/M): ');
                     READLN(RES);*)
                     CHOISIRALEAT(1,2,RES);
                     IF RES=1 THEN SUITE.COMP[1].GC:='F'
                              ELSE SUITE.COMP[1].GC:='M';

                     IF SUITE.COMP[1].GC = 'M'
                        THEN
                          BEGIN
                            (*WRITE('COMP2:FEM OU MASC ?(F/M): ');
                            READLN(RES);*)
                            CHOISIRALEAT(1,2,RES);
                            IF RES=1 THEN SUITE.COMP[2].GC:='F'
                                     ELSE SUITE.COMP[2].GC:='M';

                          END
                        ELSE SUITE.COMP[2].GC:='M'
                   END
END;(*DET2SUITE*)


PROCEDURE DET3SUITE(PERSONNE,GENRE:CHAR;VAR SUITE:SUITCOMPOSANT);
VAR RES,I:INTEGER;
BEGIN
   CASE PERSONNE OF
   '1','2':BEGIN
            FOR I:=1 TO 2 DO
              BEGIN
                (*WRITE('COMP ',I,':2 OU 3EME PERS ?(2/3) : ');
                READLN(RES);*)
                CHOISIRALEAT(2,3,RES);
                IF RES=2 THEN SUITE.COMP[I].PC:='2'
                         ELSE SUITE.COMP[I].PC:='3';

                (*WRITE('COMP ',I,':SING OU PLUR ?(S/P)    : ');
                READLN(RES);*)
                CHOISIRALEAT(1,2,RES);
                IF RES=1 THEN SUITE.COMP[I].NC:='S'
                         ELSE SUITE.COMP[I].NC:='P';

                CASE SUITE.COMP[I].PC OF
                 '3':BEGIN
                       (*WRITE('COMP',I,':PRON OU GRPNOM?(P/G) : ');
                       READLN(RES);*)
                       CHOISIRALEAT(1,2,RES);
                       IF RES=1 THEN SUITE.COMP[I].TC:='P'
                                ELSE SUITE.COMP[I].TC:='G';

                     END;
                  '2':SUITE.COMP[I].TC:='P'
                END
              END;
            (*WRITE('COMP3:SING OU PLUR ?(S/P) : ');
            READLN(RES);*)
            CHOISIRALEAT(1,2,RES);
            IF RES=1 THEN SUITE.COMP[3].NC:='S'
                     ELSE SUITE.COMP[3].NC:='P';

            SUITE.COMP[3].PC:=PERSONNE;
            SUITE.COMP[3].TC:='P'
          END;
     '3':FOR I:=1 TO 3 DO
            BEGIN
              (*WRITE('COMP',I,':SING OU PLUR ?(S/P) : ');
              READLN(RES);*)
              CHOISIRALEAT(1,2,RES);
              IF RES=1 THEN SUITE.COMP[I].NC:='S'
                       ELSE SUITE.COMP[I].NC:='P';

              (*WRITE('COMP',I,':PRON OU GRPNOM?(P/G): ');
              READLN(RES);*)
              CHOISIRALEAT(1,2,RES);
              IF RES=1 THEN SUITE.COMP[I].TC:='P'
                       ELSE SUITE.COMP[I].TC:='G';

              SUITE.COMP[I].PC:='3'
            END;
   END;
  IF GENRE='F' THEN FOR I:=1 TO 3 DO
                      SUITE.COMP[I].GC:='F'
                ELSE BEGIN
                       (*WRITELN('AU MOINS UN MASCULIN!');*)
                       FOR I:=1 TO 3 DO
                         BEGIN
                           (*WRITE('COMP ',I,'FEM OU MASC ?(F/M) : ');
                           READLN(RES);*)
                           CHOISIRALEAT(1,2,RES);
                           IF RES=1 THEN SUITE.COMP[I].GC:='F'
                                    ELSE SUITE.COMP[I].GC:='M'

                         END
                     END
END;(*DET3SUITE*)


(* DETSUITE...PRINCIPAL  *)
BEGIN
   DETNBCOMP(SUITE.NBCOMP);
   IF SUITE.NBCOMP=2
       THEN DET2SUITE(PERSONNE,GENRE,SUITE)
       ELSE DET3SUITE(PERSONNE,GENRE,SUITE)
END;(*DETSUITE*)
```

```
SEGMENT PROCEDURE CONSTRSUITE(PERSONNE,GENRE:CHAR;INDVERBE:INTEGER;
                        VAR SUITE:SUITCOMPOSANT);

VAR I:INTEGER;
    COMPOSANT:DESC;


PROCEDURE COMPGEN(VAR COMPOSANT:DESC);

 VAR FONCTION:FONC;


 PROCEDURE CONSPRONSUITE(NOMBRE,PERSONNE,GENRE:CHAR;VAR PRON:SPRONOM);
 VAR RES:INTEGER;
 BEGIN
    CASE PERSONNE OF
          '1':IF NOMBRE='S' THEN PRON:='MOI'
                            ELSE PRON:='NOUS';
          '2':IF NOMBRE='S' THEN PRON:='TOI'
                            ELSE PRON:='VOUS';
          '3':IF NOMBRE='S' THEN IF GENRE='F' THEN
                                          BEGIN
                                            CHOISIRALEAT(1,2,RES);
                                            IF RES = 1
                                               THEN PRON:='ELLE'
                                               ELSE CHOISFNPROPRE(PRON)
                                          END
                                        ELSE
                                          BEGIN
                                            CHOISIRALEAT(1,2,RES);
                                            IF RES = 1
                                               THEN PRON:='LUI'
                                               ELSE CHOISMNPROPRE(PRON)
                                          END

                              ELSE IF GENRE='F' THEN PRON:='ELLES'
                                                ELSE PRON:='EUX'
        END
 END;(*CONSPRONSUITE*)



BEGIN
   CASE COMPOSANT.TC OF
    'G':BEGIN
          FONCTION:=SUJ;
          CONSGRPNOM(COMPOSANT.NC,COMPOSANT.GC,TYPEVB,INDVERBE,FONCTION,
                     COMPOSANT.CG)
        END;
    'P':CONSPRONSUITE(COMPOSANT.NC,COMPOSANT.PC,COMPOSANT.GC,COMPOSANT.CP)
    END
END;(*COMPGEN*)



(* CONSTRSUITE...PRINCIPAL *)

BEGIN
   FOR I:=1 TO SUITE.NBCOMP DO
      BEGIN
         COMPOSANT:=SUITE.COMP[I];
         COMPGEN(COMPOSANT);
         SUITE.COMP[I]:=COMPOSANT
      END
END;(*CONSTRSUITE*)
```

```
(* CONSSUITE...PRINCIPAL*)
BEGIN
   DETSUITE(PERSONNE,GENRE,SUITE);
   CONSTRSUITE(PERSONNE,GENRE,INDVERBE,SUITE)
END;(*CONSSUITE*)


(****************************************)
```

```
SEGMENT PROCEDURE DETPHRASE(VAR DSUJET:SUJETDESC;
                            VAR NOVERBE,NG,TEMPS,PERS,PHRASETYP:INTEGER);



  PROCEDURE DETPHTYPE(VAR PHRASETYP:INTEGER);

  BEGIN
     (*WRITE('TYPE DE PHRASE ? (1/10) ');
      READLN(PHRASETYP)*)
  END;(*DETPHTYPE*)



  PROCEDURE DETNOMBRE(VAR DSUJ:SUJETDESC);

  VAR REP:CHAR;
      RES:INTEGER;

  BEGIN
     (*WRITE('SUJET SINGULIER OU PLURIEL?(S/P) : ');
     READLN(REP);*)
     CHOISIRALEAT(1,2,RES);
     IF RES = 1 THEN REP:='S'
                ELSE REP:='P';
     DSUJ.NOMBRE:=REP
  END;(*DETNOMBRE*)



  PROCEDURE DETPERSONNE(VAR DSUJ:SUJETDESC);

  VAR REP:CHAR;
      RES:INTEGER;

  BEGIN
     (*WRITE('SUJET 1,2,3EME PERSONNE?(1,2,3) : ');
     READLN(REP);*)
     CHOISIRALEAT(1,5,RES);
     CASE RES OF
      1     :REP:='1';
      2     :REP:='2';
      3,4,5:REP:='3';
      END;

     DSUJ.PERSONNE:=REP
  END;(*DETPERSONNE*)
```

```
PROCEDURE DETGENRE(VAR DSUJ:SUJETDESC);

VAR REP:CHAR;
    RES:INTEGER;

BEGIN
   IF (DSUJ.PERSONNE='3') AND (DSUJ.NOMBRE='S')
      THEN (*WRITE('SUJET FEM,MASC OU IMPERSONNEL?(F,M,I) : ')*)
        BEGIN
          CHOISIRALEAT(1,3,RES);
          CASE RES OF
           1:REP:='F';
           2:REP:='M';
           3:REP:='I';
          END;
        END
      ELSE(*WRITE('SUJET FEMININ,MASCULIN?(F,M) : ');*)
        BEGIN
          CHOISIRALEAT(1,2,RES);
          CASE RES OF
           1:REP:='F';
           2:REP:='M';
          END;
        END;
   (*READLN(REP);*)
   DSUJ.GENRE:=REP
END;(*DETGENRE*)
```

```
PROCEDURE DETTYPESUJET(VAR DSUJ:SUJETDESC);

VAR REP:CHAR;
    RES:INTEGER;

BEGIN
   CASE DSUJ.PERSONNE OF
    '1','2':IF DSUJ.NOMBRE='S'
             THEN DSUJ.VAL.TYPSUJ:='P'
             ELSE
               BEGIN
                  (*WRITELN('TYPE DU SUJET EST PRONOM PERSONNEL');
                  WRITE  ('OU SUITE DE COMPOSANTS ?(P/S) : ');
                  READLN(REP);*)
                  CHOISIRALEAT(1,3,RES);
                  CASE RES OF
                    1:DSUJ.VAL.TYPSUJ:='P';
                  2,3:DSUJ.VAL.TYPSUJ:='S';
                   END;
               END;
       '3':IF DSUJ.NOMBRE='S'
             THEN
               BEGIN
                  (*WRITELN('TYPE DU SUJET EST PRONOM PERSONNEL');
                  WRITE  ('OU GROUPE DE NOM ? (P/G) : ');
                  READLN(REP);*)
                  CHOISIRALEAT(1,2,RES);
                  CASE RES OF
                   1:DSUJ.VAL.TYPSUJ:='P';
                   2:DSUJ.VAL.TYPSUJ:='G'
                   END
               END
             ELSE
               BEGIN
                  (*WRITELN('TYPE DU SUJET EST PRONOM PERSONNEL');
                  WRITELN('OU GROUPE DE NOM');
                  WRITE  ('OU SUITE DE COMPOSANTS ?(P/G/S) : ');
                  READLN(REP);*)
                  CHOISIRALEAT(1,4,RES);
                  CASE RES OF
                     1:DSUJ.VAL.TYPSUJ:='P';
                     2:DSUJ.VAL.TYPSUJ:='G';
                   3,4:DSUJ.VAL.TYPSUJ:='S'
                   END
               END
   END
END;(*DETTYPESUJET*)
```

```
PROCEDURE DETVERBE(VAR NOVERBE,GPVERBE,TEMPS:INTEGER;PERSONNE,NOMBRE:CHAR;
                   VAR PERS:INTEGER);
VAR NO:INTEGER;

BEGIN
  (*WRITE('NO GROUPE D''UN VERBE <1/2/3>:');
  READLN(GPVERBE);
  WRITE('NO VERBE DANS CE GROUPE');
  CASE NG OF
   1:WRITE(' MAX 21 :');
   2:WRITE(' MAX  7 :');
   3:WRITE(' MAX 16 :')
  END;
  READLN(NOVERBE);
  WRITELN('A QUEL TEMPS ?');
  WRITELN('   PRESENT              [1]');
  WRITELN('   IMPARFAIT            [2]');
  WRITELN('   PASSE SIMPLE         [3]');
  WRITELN('   FUTUR SIMPLE         [4]');
  WRITELN('   PASSE COMPOSE        [5]');
  WRITELN('   PLUS-QUE-PARFAIT     [6]');
  WRITELN('   PASSE ANTERIEUR      [7]');
  WRITELN('   FUTUR ANTERIEUR      [8]');
  WRITE('   CHOIX :             ');
  READLN(TEMPS);
  CASE PERSONNE OF
   '1':IF NOMBRE = 'S' THEN PERS:=1
                       ELSE PERS:=4;
   '2' :IF NOMBRE = 'S' THEN PERS:=2
                        ELSE PERS:=5;
   '3' :IF NOMBRE = 'S' THEN PERS:=3
                        ELSE PERS:=6
  END;*)

  CHOISIRALEAT(1,44,NO);
  IF NO<=21 THEN
              BEGIN
                GPVERBE:=1;
                CHOISIRALEAT(1,21,NOVERBE)
              END
            ELSE IF NO<=28
                   THEN
                     BEGIN
                       GPVERBE:=2;
                       CHOISIRALEAT(1,7,NOVERBE)
                     END
                   ELSE
                     BEGIN
                       GPVERBE:=3;
                       CHOISIRALEAT(1,16,NOVERBE)
                     END;

  CHOISIRALEAT(1,8,TEMPS);

  CASE PERSONNE OF
   '1':IF NOMBRE='S' THEN PERS:=1
                     ELSE PERS:=4;
   '2':IF NOMBRE='S' THEN PERS:=2
                     ELSE PERS:=5;
   '3':IF NOMBRE='S' THEN PERS:=3
                     ELSE PERS:=6
  END
END;(*DETVERBE*)
```

```
BEGIN
   DETPHTYP(PHRASETYP);
   DETNOMBRE(DSUJET);
   DETPERSONNE(DSUJET);
   DETGENRE(DSUJET);
   DETTYPESUJET(DSUJET);
   DETVERBE(NOVERBE,NG,TEMPS,DSUJET.PERSONNE,DSUJET.NOMBRE,PERS)
END;(*DETPHRASE*)



PROCEDURE CHARBESCH;

(* UN GRAND MOT POUR UN PETIT CHIPOTAGE. *)

BEGIN
  LONGUEUR:=LONGUEUR+1;
  MOVELEFT(TERMINAISON[SOURCE-1],DESTINATION,LONGUEUR);
  LONGUEUR:=LONGUEUR-1;
  MOVELEFT(LONGUEUR,DESTINATION,1)
END;(*CHARBESCH*)



PROCEDURE CHARBYTE;

(* UN GRAND MOT POUR UN PETIT CHIPOTAGE. *)

BEGIN
  LONGUEUR:=LONGUEUR+1;
  MOVELEFT(LISTMOT[SOURCE-1],DESTINATION,LONGUEUR);
  LONGUEUR:=LONGUEUR-1;
  MOVELEFT(LONGUEUR,DESTINATION,1)
END;(*CHARBYTE*)



PROCEDURE CHOISIRALEAT;

BEGIN
  (*WRITE('NOMBRE ALEATOIRE [',BI,',',BS,'] : ');
  READLN(RESULT)*)
  RESULT:=BI + RANDOM MOD (BS-BI+1)
END;(*CHOISIRALEAT*)



PROCEDURE PREMLETTRE;

BEGIN
  LETTRE:=MOT[1]
END;(*PREMLETTRE*)
```

```
PROCEDURE CHOISADV;

VAR IND,ADRMOT,LONGMOT,RES:INTEGER;

BEGIN
  IF LGLISTE = 0 THEN WRITE('ERREUR:CHOISIADV LGLISTE = 0')
                 ELSE CHOISIRALEAT(1,LGLISTE,RES);
  IND:=TADV[LISTE[RES]];
  ADRMOT:=INDICE[IND].ADRESSE;
  LONGMOT:=INDICE[IND+1].ADRESSE-ADRMOT-1;
  CHARBYTE(ADRMOT,LONGMOT,MOT)
END;(*CHOISADV*)



PROCEDURE CHOISFNPROPRE;

VAR IND,ADRMOT,LONGMOT,RES:INTEGER;

BEGIN
  CHOISIRALEAT(1,NUMFNP,RES);
  IND:=TFNPROP[RES];
  ADRMOT:=INDICE[IND].ADRESSE;
  LONGMOT:=INDICE[IND+1].ADRESSE-ADRMOT-1;
  CHARBYTE(ADRMOT,LONGMOT,MOT)
END;(*CHOISFNPROPRE*)



PROCEDURE CHOISMNPROPRE;

VAR IND,ADRMOT,LONGMOT,RES:INTEGER;

BEGIN
  CHOISIRALEAT(1,NUMMNP,RES);
  IND:=TMNPROP[RES];
  ADRMOT:=INDICE[IND].ADRESSE;
  LONGMOT:=INDICE[IND+1].ADRESSE-ADRMOT-1;
  CHARBYTE(ADRMOT,LONGMOT,MOT)
END;(*CHOISFNPROPRE*)



PROCEDURE ACCEDE(NG,TP,NP:INTEGER;VAR RES:STRING);

VAR LONGUEUR,IND:INTEGER;

BEGIN
  IF TP = 5 THEN IND:=NG*25
            ELSE IND:=(NG-1)*25+(TP-1)*6+NP;
  LONGUEUR:=INDIRECT[IND+1]-INDIRECT[IND];
  CHARBESCH(INDIRECT[IND],LONGUEUR,RES)
END;(*ACCEDE*)
```

```
PROCEDURE ECRIREVERBE(TEMPS:INTEGER;VAR AUX,VERBCONJ:STRING);
VAR TAILLE:INTEGER;

BEGIN
  IF TEMPS<=4
    THEN
      BEGIN
        TAILLE:=LENGTH(VERBCONJ);
        PASSAGEALALIGNE(LONGLIGNE,TAILLE);
        WRITE(VERBCONJ);
        BLANC(LONGLIGNE)
      END
    ELSE
      BEGIN
        TAILLE:=LENGTH(AUX);
        PASSAGEALALIGNE(LONGLIGNE,TAILLE);
        WRITE(AUX);
        BLANC(LONGLIGNE);
        TAILLE:=LENGTH(VERBCONJ);
        PASSAGEALALIGNE(LONGLIGNE,TAILLE);
        WRITE(VERBCONJ);
        BLANC(LONGLIGNE)
      END
END;(*ECRIREVERBE*)



PROCEDURE ECRPRONOM(PRONOM:SPRONOM);

VAR TAILLE:INTEGER;

BEGIN
   TAILLE:=LENGTH(PRONOM);
   PASSAGEALALIGNE(LONGLIGNE,TAILLE);
   WRITE(PRONOM)
END;(*ECRPRONOM*)
```

```
PROCEDURE CONJVERBE (NOVERBE,GPVERBE,TEMPS,PERS:INTEGER;
                     VAR AUX,VERBCONJ:STRING;VAR NEXTLETTRE:CHAR);

VAR LONGUEUR,NO,NG:INTEGER;
    TER:STRING;

BEGIN
  CASE GPVERBE OF
   1:BEGIN
       NG:=TEVERBE[NOVERBE].TERMVB.CONJ;
       NO:=TEVERBE[NOVERBE].ADIND;
     END;
   2:BEGIN
       NG:=TISVERBE[NOVERBE].TERMVB.CONJ;
       NO:=TISVERBE[NOVERBE].ADIND;
     END;
   3:BEGIN
       NG:=TSVERBE[NOVERBE].TERMVB.CONJ;
       NO:=TSVERBE[NOVERBE].ADIND;
     END;
  END;
  LONGUEUR:=INDICE[NO+1].ADRESSE-INDICE[NO].ADRESSE-1;
  CHARBYTE(INDICE[NO].ADRESSE,LONGUEUR,VERBCONJ);
  IF TEMPS <=4
     THEN BEGIN
            ACCEDE(NG,TEMPS,PERS,TER);
            VERBCONJ:=CONCAT(VERBCONJ,TER);
            PREMLETTRE(VERBCONJ,NEXTLETTRE)
          END
     ELSE BEGIN
            ACCEDE(1,TEMPS-4,PERS,AUX);
            ACCEDE(NG,5,PERS,TER);
            VERBCONJ:=CONCAT(VERBCONJ,TER);
            PREMLETTRE(AUX,NEXTLETTRE)
          END;
END;(*CONJVERBE*)



PROCEDURE PASSAGEALALIGNE;

BEGIN
  IF LONGLIGNE+LONGAJOUTEE > 40
     THEN
       BEGIN
         WRITELN;
         LONGLIGNE:=LONGAJOUTEE;
       END
     ELSE LONGLIGNE:=LONGLIGNE+LONGAJOUTEE
END;(*PASSAGEALALIGNE*)



PROCEDURE BLANC;

BEGIN
  IF LONGLIGNE <> 0
     THEN
        BEGIN
          WRITE(' ');
          LONGLIGNE:=LONGLIGNE+1
        END
END;(*BLANC*)
```

```
PROCEDURE ECRGRNOM(GRP:GRNOM);
VAR TAILLE:INTEGER;
BEGIN
   IF GRP.INDIC[1]<>0
         THEN
           BEGIN
             TAILLE:=LENGTH(GRP.DETERMINANT.D1);
             PASSAGEALALIGNE(LONGLIGNE,TAILLE);
             WRITE(GRP.DETERMINANT.D1);
             BLANC(LONGLIGNE)
           END;

   IF GRP.INDIC[2]<>0
         THEN
           BEGIN
             TAILLE:=LENGTH(GRP.DETERMINANT.D2);
             PASSAGEALALIGNE(LONGLIGNE,TAILLE);
             WRITE(GRP.DETERMINANT.D2);
             BLANC(LONGLIGNE)
           END;


   IF GRP.INDIC[3]<>0
         THEN
           BEGIN
             TAILLE:=LENGTH(GRP.ADAV);
             PASSAGEALALIGNE(LONGLIGNE,TAILLE);
             WRITE(GRP.ADAV);
             BLANC(LONGLIGNE)
           END;

   IF GRP.INDIC[4]<>0
         THEN
           BEGIN
             TAILLE:=LENGTH(GRP.NOM);
             PASSAGEALALIGNE(LONGLIGNE,TAILLE);
             WRITE(GRP.NOM);
             BLANC(LONGLIGNE)
           END;

   IF GRP.INDIC[5]<>0
         THEN
           BEGIN
             TAILLE:=LENGTH(GRP.ADAP);
             PASSAGEALALIGNE(LONGLIGNE,TAILLE);
             WRITE(GRP.ADAP);
             BLANC(LONGLIGNE)
           END;
END;(*ECRGRNOM*)
```

```
PROCEDURE ECRSUITE(SUITE:SUITCOMPOSANT);

VAR TAILLE,I,RES:INTEGER;
    INTER:DESC;
    REPONSE:BOOLEAN;


 PROCEDURE TOUSSINGULIER(SUITE:SUITCOMPOSANT;VAR REPONSE:BOOLEAN);
 VAR I:INTEGER;

 BEGIN
   I:=1;
   REPONSE:=TRUE;
   WHILE (I<=SUITE.NBCOMP) AND (REPONSE = TRUE) DO
       IF SUITE.COMP[I].NC = 'P'
          THEN REPONSE:=FALSE
          ELSE I:=I+1
 END;(*TOUSSINGULIER*)


BEGIN
   FOR I:=1 TO SUITE.NBCOMP DO
     BEGIN
       INTER:=SUITE.COMP[I];
       IF INTER.TC='P' THEN
                          BEGIN
                            TAILLE:=LENGTH(INTER.CP);
                            PASSAGEALALIGNE(LONGLIGNE,TAILLE);
                            WRITE(INTER.CP);
                          END
                        ELSE ECRGRNOM(INTER.CG);
       IF (SUITE.NBCOMP-1)=I
                        THEN
                          BEGIN
                            TOUSSINGULIER(SUITE,REPONSE);
                            IF REPONSE
                               THEN RES:=1
                               ELSE CHOISIRAL(1,2,RES);
                            IF RES=1 THEN
                                      BEGIN
                                        BLANC(LONGLIGNE);
                                        TAILLE:=2;
                                        PASSAGEALALIGNE(LONGLIGNE,TAILLE);
                                        WRITE('ET');
                                        BLANC(LONGLIGNE)
                                      END

                                    ELSE
                                      BEGIN
                                        BLANC(LONGLIGNE);
                                        TAILLE:=2;
                                        PASSAGEALALIGNE(LONGLIGNE,TAILLE);
                                        WRITE('OU');
                                        BLANC(LONGLIGNE)
                                      END

                        END
                      ELSE IF(SUITE.NBCOMP<>I)
                                  THEN
                                    BEGIN
                                      TAILLE:=1;
                                      PASSAGEALALIGNE(LONGLIGNE,TAILLE);
                                      WRITE(',')
                                    END
                                  ELSE BLANC(LONGLIGNE)

     END
END;(*ECRSUITE*)
```

```
PROCEDURE CONSTRPROPERS(VAR SJ:SUJETDESC;NEXTLETTRE:CHAR);
VAR RES:INTEGER;

BEGIN
   CASE SJ.NOMBRE OF
    'S':CASE SJ.PERSONNE OF
        '1':BEGIN
             SJ.VAL.PRONOM:='JE ';
             CASE NEXTLETTRE OF
             'A','E','I','O','U','Y':SJ.VAL.PRONOM:='J'';
             END
            END;

        '2':SJ.VAL.PRONOM:='TU ';
        '3':BEGIN
             CASE SJ.GENRE OF
                  'F':BEGIN
                      CHOISIRALEAT(1,2,RES);
                      IF RES = 1 THEN SJ.VAL.PRONOM:='ELLE '
                                 ELSE CHOISMNPROPRE(SJ.VAL.PRONOM)
                      END;

                  'M':BEGIN
                      CHOISIRALEAT(1,2,RES);
                      IF RES = 1 THEN SJ.VAL.PRONOM:='IL '
                                 ELSE CHOISMNPROPRE(SJ.VAL.PRONOM)
                      END;

                  'I':SJ.VAL.PRONOM:='ON ';
                  END;
             SJ.VAL.PRONOM:=CONCAT(SJ.VAL.PRONOM,' ')
            END;

        END;
    'P':CASE SJ.PERSONNE OF
        '1':SJ.VAL.PRONOM:='NOUS ';
        '2':SJ.VAL.PRONOM:='VOUS ';
        '3':CASE SJ.GENRE OF
            'F':SJ.VAL.PRONOM:='ELLES ';
            'M':SJ.VAL.PRONOM:='ILS '
            END
        END
    END
END;(*CONSTRPROPERS*)



PROCEDURE GENSUJ(VAR DSUJET:SUJETDESC;TYPEVB,INDVERBE:INTEGER;NEXTLETTRE:CHAR);

VAR FONCTION:FONC;

BEGIN
  FONCTION:= SUJ;
  CASE DSUJET.VAL.TYPSUJ OF
   'P':CONSTRPROPERS(DSUJET,NEXTLETTRE);
   'G':CONSGRPNOM(DSUJET.NOMBRE,DSUJET.GENRE,
                  TYPEVB,INDVERBE,FONCTION,DSUJET.VAL.GR);
   'S':CONSSUITE(DSUJET.PERSONNE,DSUJET.GENRE,TYPEVB,INDVERBE,DSUJET.VAL.ST)
  END
END;(*GENSUJ*)
```

```
PROCEDURE ECRIRESUJET(DSUJET:SUJETDESC);

BEGIN
  CASE DSUJET.VAL.TYPSUJ OF
  'P':ECRPRONOM(DSUJET.VAL.PRONOM);
  'G':ECRGRNOM(DSUJET.VAL.GR);
  'S':ECRSUITE(DSUJET.VAL.ST)
  END
END;(*ECRIRESUJET*)



PROCEDURE GENCODCOIA(VAR COMPL:GRNOM;VAR NOMBRE:CHAR;TYPEVB,INDVERBE:INTEGER;
                     VAR FONCTION:FONC);
VAR RES:INTEGER;
    GENRE:CHAR;

BEGIN
  CHOISIRALEAT(1,2,RES);
  IF RES=1 THEN FONCTION:=COD
           ELSE FONCTION:=COIA;
  CHOISIRALEAT(1,2,RES);
  IF RES = 1 THEN NOMBRE:='S'
             ELSE NOMBRE:='P';
  CHOISIRALEAT(1,2,RES);
  IF RES = 1 THEN GENRE:='F'
             ELSE GENRE:='M';
  CONSGRPNOM(NOMBRE,GENRE,TYPEVB,INDVERBE,FONCTION,COMPL)
END;(*GENCODCOIA*)
```

```
PROCEDURE ECRIRECOIA(VAR COIA:GRNOM;NOMBRE:CHAR);

VAR TAILLE,RES:INTEGER;
    MOTIMP:STRING[3];


BEGIN
  MOTIMP:='VID';
  IF NOMBRE = 'S'
     THEN
       CASE COIA.INDIC[1] OF
            1:IF COMPL.DETERMINANT.D1='LE'
                      THEN COIA.DETERMINANT.D1:='AU'
                      ELSE MOTIMP:='A';

         2,3,4:MOTIMP:='A';
       END
     ELSE
       CASE COIA.INDIC[1] OF
            0:IF COIA.INDIC[2]=1 THEN
                                  BEGIN
                                    CHOISIRALEAT(1,2,RES);
                                    IF RES=1
                                       THEN MOTIMP:='A'
                                       ELSE MOTIMP:='AUX'
                                  END;
            1:COIA.DETERMINANT.D1:='AUX';
         2,3,4:MOTIMP:='A'
       END;


  IF MOTIMP <> 'VID'
     THEN
       BEGIN
         TAILLE:=LENGTH(MOTIMP);
         PASSAGEALALIGNE(LONGLIGNE,TAILLE);
         WRITE(MOTIMP);
         BLANC(LONGLIGNE)
       END;

  ECRGRNOM(COIA);

END;(*ECRIRECOIA*)
```

```
PROCEDURE ECRCODCOIA(VAR COMPL:GRNOM;FONCTION:FONC;NOMBRE:CHAR);

VAR TAILLE:INTEGER;

BEGIN
  CASE FONCTION OF

    COD   :ECRGRNOM(COMPL);
    COIA  :ECRIRECOIA(COMPL,NOMBRE);
    ADV   :BEGIN
             TAILLE:=LENGTH(COMPL.NOM);
             PASSAGEALALIGNE(LONGLIGNE,TAILLE);
             WRITE(COMPL.NOM)
           END
  END
END;(*ECRCODCOIA*)
```

```
(* TEXTE DU PROGRAMME PRINCIPAL ... *)



BEGIN
  PAGE(OUTPUT);
  CHARGER;
  FINI:=FALSE;
  WHILE NOT FINI DO (* CE PROGRAMME BOUCLE SANS FIN *)


    BEGIN
      LONGLIGNE:=1;
      RANDOMIZE;
      DETPHRASE(DSUJET,NOVERBE,NG,TEMPS,PERS,PHRASETYP);

      CONJVERBE(NOVERBE,NG,TEMPS,PERS,AUX,VERBCONJ,NEXTLETTRE);
      GENSUJ(DSUJET,NG,NOVERBE,NEXTLETTRE);
      GENCODCOIA(COMPL,NBRCOI,NG,NOVERBE,FONCCPL);

      WRITE(' ');
      ECRIRESUJET(DSUJET);
      ECRIREVERBE(TEMPS,AUX,VERBCONJ);
      ECRCODCOIA(COMPL,FONCCPL,NBRCOI);
      WRITELN;
      WRITELN;
      WRITELN;

    END;
END.(*GENERATION*)
```

```
(*$S+*)

PROGRAM CORMOT;



(* SYSTEME DE CORRECTION AUTOMATIQUE DES

   FAUTES D'ORTHOGRAPHE D'USAGE ET DES

   FAUTES DE FRAPPE                          *)



CONST NBREMOT  = 300;
      NBRECAR  = 2200;
      NBRETERM = 100;



TYPE MATRICE = ARRAY ['@'..'Z','@'..'Z'] OF INTEGER;
     STR1    = STRING[1];
     CHAINE  = STRING[20];
     TABCAR  = PACKED ARRAY[1..NBRECAR] OF CHAR;
     TAB7    = ARRAY [1..7] OF INTEGER;
     TAB6    = ARRAY [1..6] OF INTEGER;
     TAB26   = ARRAY [1..27] OF CHAINE;
     TABIND  = ARRAY[1..NBREMOT] OF INTEGER;
     TABTERM = ARRAY[1..NBRETERM]OF INTEGER;
     NATURE  = (FNPROPRE,MNPROPRE,NCOM,EVERBE,ISVERBE,SVERBE,ADJQ,ADVMAN);
     WORD    = PACKED RECORD
                  ORTHO:STRING[20];
                  NAT:NATURE
               END;



VAR REPUTIL:CHAR;
    BI,BS,DIMMOT,DIMCAR,DIMTERM,IND,COMMAX,POSDEPART,FREQNUL:INTEGER;
    ENTREE:CHAINE;
    ORTHO:BOOLEAN;
    FREQUENCE:TAB7;
    POSLETCRIT,PROD:TAB6;
    DIMPROD,DIMFREQ,DIMPOSLETCRIT:INTEGER;
    DIGRAM:MATRICE;
    INDICE:TABIND;
    INDICETERM:TABTERM;
    LISTMOT:TABCAR;
    FICHPREFIXES:FILE OF WORD;
    CORSTR:PACKED ARRAY ['@'..'Z'] OF STR1;
```

```
PROCEDURE CHARGEMENT;

VAR IND,IND3,IND4,IND5,DEBTERMENT,INDTERMENT:INTEGER;
    TERMENT:STRING[3];
    TABTERM:ARRAY[1..NBREMOT]OF STRING[3];


BEGIN
  PAGE(OUTPUT);
  WRITELN('       UNE PETITE MINUTE DE SILENCE          ');
  IND3:=0;
  IND:=1;
  TABTERM[1]:='ZZZ';
  LISTMOT[IND]:='@';
  DIMTERM:=0;
  RESET(FICHPREFIXES,'#5:TEST2.DATA');
  WHILE NOT EOF(FICHPREFIXES) DO
     BEGIN
       IND3:=IND3+1;
       IND:=IND+1;
       INDICE[IND3]:=IND;
       ENTREE:=FICHPREFIXES^.ORTHO;
       MOVELEFT(ENTREE[1],LISTMOT[IND],LENGTH(ENTREE));
       IND:=IND+LENGTH(ENTREE);
       IF LENGTH(ENTREE) >6 THEN
         BEGIN
           DEBTERMENT:=LENGTH(ENTREE)-2;
           TERMENT:=COPY(ENTREE,DEBTERMENT,3);
           INDTERMENT:=IND-3;
           IND4:=1;
           WHILE ((IND4<=DIMTERM)AND(TERMENT>=TABTERM[IND4])) DO IND4:=IND4+1;
           DIMTERM:=DIMTERM+1;
           IF DIMTERM>1 THEN FOR IND5:=DIMTERM DOWNTO IND4+1 DO
             BEGIN
               TABTERM[IND5]:=TABTERM[IND5-1];
               INDICETERM[IND5]:=INDICETERM[IND5-1]
             END;
           TABTERM[IND4]:=TERMENT;
           INDICETERM[IND4]:=INDTERMENT
         END;
       LISTMOT[IND]:='@';
       GET(FICHPREFIXES)
     END;
  INDICE[IND3+1]:=IND+1;
  LISTMOT[IND+1]:='Z';
  LISTMOT[IND+2]:='Z';
  LISTMOT[IND+3]:='@';
  DIMCAR:=IND+3;
  DIMMOT:=IND3+1;
  CLOSE(FICHPREFIXES,LOCK)
END;(*CHARGEMENT*)
```

```
PROCEDURE CORSTRING;

VAR I:CHAR;


BEGIN
     CORSTR['@']:='@';
     CORSTR['A']:='A';
     CORSTR['B']:='B';
     CORSTR['C']:='C';
     CORSTR['D']:='D';
     CORSTR['E']:='E';
     CORSTR['F']:='F';
     CORSTR['G']:='G';
     CORSTR['H']:='H';
     CORSTR['I']:='I';
     CORSTR['J']:='J';
     CORSTR['K']:='K';
     CORSTR['L']:='L';
     CORSTR['M']:='M';
     CORSTR['N']:='N';
     CORSTR['O']:='O';
     CORSTR['P']:='P';
     CORSTR['Q']:='Q';
     CORSTR['R']:='R';
     CORSTR['S']:='S';
     CORSTR['T']:='T';
     CORSTR['U']:='U';
     CORSTR['V']:='V';
     CORSTR['W']:='W';
     CORSTR['X']:='X';
     CORSTR['Y']:='Y';
     CORSTR['Z']:='Z';
END;(*CORSTRING*)



PROCEDURE CONSMATDIG;

VAR LIGNE,COLONNE:CHAR;
    I:INTEGER;


BEGIN
  FOR LIGNE:= '@' TO 'Z' DO
    FOR COLONNE:= '@' TO 'Z' DO
      DIGRAM[LIGNE,COLONNE]:=0;
  FOR I:=1 TO DIMCAR-4 DO
    DIGRAM[LISTMOT[I],LISTMOT[I+1]]:=DIGRAM[LISTMOT[I],LISTMOT[I+1]] + 1
END;(*CONSMATDIG*)



PROCEDURE CHARBYTE(SOURCE:INTEGER;VAR LONGUEUR:INTEGER;VAR DESTINATION:CHAINE);

BEGIN
  LONGUEUR:=LONGUEUR+1;
  MOVELEFT(LISTMOT[SOURCE-1],DESTINATION,LONGUEUR);
  LONGUEUR:=LONGUEUR-1;
  MOVELEFT(LONGUEUR,DESTINATION,1)
END;(*CHARBYTE*)
```

```
PROCEDURE COMPARE(ENTREE,FICHMOT:CHAINE;VAR COMCOUR:INTEGER);

VAR TROUVE:BOOLEAN;
    IND:INTEGER;


BEGIN
  TROUVE:=TRUE;
  IND:=1;
  WHILE((IND<=LENGTH(ENTREE))AND(IND<=(LENGTH(FICHMOT)))AND TROUVE)
    DO IF ENTREE[IND] = FICHMOT[IND]
      THEN IND:=IND+1
      ELSE TROUVE:=FALSE;
  COMCOUR:=IND-1
END;(*COMPARE*)



PROCEDURE RECHERCHE (ENTREE:CHAINE;VAR ORTHO:BOOLEAN;VAR BI,BS:INTEGER);

VAR DEMI,INDCAR,LONGUEUR:INTEGER;
    INTER:CHAINE;
    CONTINUE:BOOLEAN;


BEGIN
  ORTHO:=FALSE;
  CONTINUE:=TRUE;
  BI:=1;
  BS:=DIMMOT-1;
  WHILE (CONTINUE)AND(BI<=BS) DO
    BEGIN
      DEMI:=(BI+BS)DIV 2;
      INDCAR:=INDICE[DEMI];
      LONGUEUR:=INDICE[DEMI+1]-INDCAR-1;
      CHARBYTE(INDCAR,LONGUEUR,INTER);
      IF ENTREE=INTER
            THEN
             BEGIN
              CONTINUE:=FALSE;
              ORTHO:=TRUE
             END
            ELSE IF ENTREE<INTER
                  THEN BS:=DEMI-1
                  ELSE BI:=DEMI+1
    END
END;(*RECHERCHE*)
```

```
PROCEDURE CONTCARCOM(BI,BS:INTEGER;ENTREE:CHAINE;
                     VAR COMMAX,POSDEPART:INTEGER);
VAR COMCOUR:INTEGER;


 PROCEDURE COMPMOT(NO:INTEGER;ENTREE:CHAINE;VAR COMMUNE:INTEGER);

 (* CETTE PROCEDURE COMPARE 'ENTREE' ET
    LE MOT DE NO 'NUMERO', 'COMMUNE EST
    LE NOMBRE DE 1ERES LETTRES QU'ILS
    ONT EN COMMUN                          *)

 VAR MOT:CHAINE;
     LONGUEUR:INTEGER;
 BEGIN
    LONGUEUR:=INDICE[NO+1]-INDICE[NO]-1;
    CHARBYTE(INDICE[NO],LONGUEUR,MOT);
    COMPARE(ENTREE,MOT,COMMUNE)
 END;(*COMPMOT*)


BEGIN
  COMCOUR:=0;
  COMMAX:=0;
  IF BS>0 THEN COMPMOT(BS,ENTREE,COMCOUR);
  IF BI<(DIMMOT-1)THEN
                    BEGIN
                      COMPMOT(BI,ENTREE,COMMAX);
                      POSDEPART:=BI
                    END;
  IF COMCOUR>COMMAX
     THEN
        BEGIN
           COMMAX:=COMCOUR;
           POSDEPART:=BS
        END
     ELSE POSDEPART:=BI
END;(*CONTCARCOM*)
```

```
PROCEDURE CALPRODIGMOT(ENTREE:CHAINE;VAR FREQUENCE:TAB7;VARPROD:TAB6;
                       VAR DIMFREQ,DIMPROD,FREQNUL:INTEGER);

VAR PREMIND,DEUXIND:CHAR;
    TRAVAIL:CHAINE;
    CONTINUE:BOOLEAN;
    LGTRAV,IND:INTEGER;


BEGIN
  IF(LENGTH(ENTREE)=COMMAX) OR (LENGTH(ENTREE)=COMMAX+1)
    THEN
       BEGIN
          TRAVAIL:=CONCAT('@',ENTREE,'@');
          LGTRAV:=LENGTH(TRAVAIL)
       END
    ELSE
       BEGIN
          TRAVAIL:=CONCAT('@',ENTREE);
          LGTRAV:=COMMAX+3
       END;

  CONTINUE:=TRUE;
  FREQNUL:=0;
  IND:=1;
  DIMPROD:=0;
  DIMFREQ:=0;

  WHILE (IND<LGTRAV) AND CONTINUE DO
    BEGIN
      PREMIND:=TRAVAIL[IND];
      DEUXIND:=TRAVAIL[IND+1];
      DIMFREQ:=DIMFREQ+1;
      FREQUENCE[DIMFREQ]:=DIGRAM[PREMIND,DEUXIND];
      IF DIMFREQ>1
         THEN
           BEGIN
              DIMPROD:=DIMPROD+1;
              PROD[DIMPROD]:=FREQUENCE[DIMFREQ-1]*FREQUENCE[DIMFREQ]
           END;
      IF FREQUENCE[DIMFREQ]=0
       THEN
         BEGIN
           IF IND <> LGTRAV-1
              THEN
                BEGIN
                   DIMPROD:=DIMPROD+1;
                   PROD[DIMPROD]:=0
                END;
           FREQNUL:=IND;
           CONTINUE:=FALSE
         END;
      IND:=IND + 1;
    END
END;(*CALPRODIGMOT*)
```

```
PROCEDURE RECHMINPROD(PROD:TAB6;DIMPROD,FREQNUL:INTEGER;
                      VAR POSLETCRIT:TAB6;VAR DIMPOSLETCRIT:INTEGER);

VAR IND1,IND2,MIN,PM:INTEGER;


BEGIN
  IF FREQNUL = 0
    THEN
      BEGIN
        FOR IND1:=1 TO DIMPROD DO
          BEGIN
            MIN:=PROD[1];
            PM:=1;
            FOR IND2:=2 TO DIMPROD DO
              IF PROD[IND2] <= MIN THEN
                  BEGIN
                    MIN:=PROD[IND2];
                    PM:=IND2
                  END;
            PROD[PM]:=MAXINT;
            POSLETCRIT[IND1]:=PM
          END;
        DIMPOSLETCRIT:=DIMPROD
      END
    ELSE
      IF FREQNUL = 1
         THEN
           BEGIN
             POSLETCRIT[1]:=1;
             DIMPOSLETCRIT:=1
           END
         ELSE IF (PROD[FREQNUL-1]=0)AND(PROD[FREQNUL]=0)
              THEN
                BEGIN
                  POSLETCRIT[1]:=FREQNUL-1;
                  POSLETCRIT[2]:=FREQNUL;
                  DIMPOSLETCRIT:=2
                END
              ELSE
                BEGIN
                  POSLETCRIT[1]:=FREQNUL-1;
                  DIMPOSLETCRIT:=1
                END;
END;(*RECHMINPROD*)
```

```
PROCEDURE SUPPRESSION(POSLETCRIT:TAB6;DIMPOSLETCRIT:INTEGER;
                      VAR ENTREE:CHAINE;VAR ORTHO:BOOLEAN);

VAR IND,POSCRIT,BIDON,BIDON1:INTEGER;
    TRAVAIL:CHAINE;


BEGIN
  IND:=1;
  TRAVAIL:=ENTREE;
  WHILE (IND<=DIMPOSLETCRIT)AND(NOT ORTHO)DO
    BEGIN
      POSCRIT:=POSLETCRIT[IND];
      DELETE(TRAVAIL,POSCRIT,1);
      RECHERCHE(TRAVAIL,ORTHO,BIDON1,BIDON);
      IF NOT ORTHO THEN TRAVAIL:=ENTREE
                   ELSE ENTREE:=TRAVAIL;
      IND:=IND+1
    END
END;(*SUPPRESsION*)



(*$I #5:CORMOT2.TEXT  *)


(* CORMOT2 *)


PROCEDURE REMPLACLETTRE(POSCRIT:INTEGER;VAR ENTREE:CHAINE;VAR CHOISIE:CHAINE);

BEGIN
  DELETE (ENTREE,POSCRIT,1);
  INSERT (CHOISIE,ENTREE,POSCRIT)
END;(*REMPLACLETTRE*)
```

```
PROCEDURE CLASLETREMP(LETAV,LETAP:CHAR;VAR LETREMP:TAB26;
                      VAR COMPTLET:INTEGER);

TYPE TAB126 = ARRAY [1..26] OF INTEGER;


VAR VALREMP:TAB126;
    MINCOUR:INTEGER;
    VAL:INTEGER;
    LET:CHAR;
    DERVAL:INTEGER;


 PROCEDURE INSERER(VAL:INTEGER;LET:STR1;VAR VALREMP:TAB126;
                   VAR LETREMP:TAB26;VAR MINCOUR:INTEGER);
 VAR I:INTEGER;


    PROCEDURE MOVLETVAL(DEB,FIN:INTEGER;VAR LETREMP:TAB26;VAR VALREMP:TAB126);

    VAR I:INTEGER;
    BEGIN
      FOR I:= FIN DOWNTO DEB DO
          BEGIN
            LETREMP[I+1]:=LETREMP[I];
            VALREMP[I+1]:=VALREMP[I]
          END
    END;(*MOVLETVAL*)



 BEGIN
   I:=1;
   WHILE VALREMP[I] > VAL DO I:=I+1;
   MOVLETVAL(I,MINCOUR-1,LETREMP,VALREMP);
   LETREMP[I]:=LET;
   VALREMP[I]:=VAL;
   MINCOUR:=MINCOUR+1
 END;(*INSERER*)



BEGIN
   DERVAL:=MAXINT;
   MINCOUR:=1;
   FOR LET:='A' TO 'Z' DO
     BEGIN
       VAL:=DIGRAM[LETAV,LET]*DIGRAM[LET,LETAP];
       IF VAL<>0
         THEN
           IF VAL <= DERVAL
           THEN
             BEGIN
               DERVAL:=VAL;
               VALREMP[MINCOUR]:=VAL;
               LETREMP[MINCOUR]:=CORSTR[LET];
               MINCOUR:=MINCOUR+1
             END
           ELSE INSERER(VAL,CORSTR[LET],VALREMP,LETREMP,MINCOUR)
     END;
   COMPTLET:=MINCOUR-1
END;(*CLASLETREMP*)
```

```
PROCEDURE REMPLACEMENT(POSLETCRIT:TAB6;DIMPOSLETCRIT:INTEGER;
                       VAR ENTREE:CHAINE;VAR ORTHO:BOOLEAN);

VAR IND,IND2,BIDON1,BIDON2,COMPTLET,POSCRIT:INTEGER;
    LETREMP:TAB26;
    LETAV,LETAP,LETCA:CHAR;
    TRAVAIL,REMP:CHAINE;


 PROCEDURE LETAVAP(ENTREE:CHAINE;POSCRIT:INTEGER;
                   VAR LETAV:CHAR;VAR LETAP:CHAR);
 BEGIN
      IF POSCRIT=1 THEN LETAV:='@'
                   ELSE LETAV:=ENTREE[POSCRIT-1];
      IF POSCRIT=LENGTH(ENTREE) THEN LETAP:='@'
                                ELSE LETAP:=ENTREE[POSCRIT+1]
 END;(*LETAVAP*)



BEGIN
  IND:=1;
  TRAVAIL:=ENTREE;
  WHILE (IND<=DIMPOSLETCRIT) AND (NOT ORTHO) DO
   BEGIN
    POSCRIT:=POSLETCRIT[IND];
    LETCA:=TRAVAIL[POSCRIT];
    LETAVAP(TRAVAIL,POSCRIT,LETAV,LETAP);
    CLASLETREMP(LETAV,LETAP,LETREMP,COMPTLET);
    IND2:=1;
    WHILE (IND2<=COMPTLET) AND (NOT ORTHO) DO
     BEGIN
      REMP:=LETREMP[IND2];
      IF (REMP<>CORSTR[LETCA]) THEN
       BEGIN
         REMPLACLETTRE(POSCRIT,TRAVAIL,REMP);
         RECHERCHE(TRAVAIL,ORTHO,BIDON1,BIDON2);
         IF (NOT ORTHO) THEN TRAVAIL:=ENTREE
                        ELSE ENTREE:=TRAVAIL
       END;
      IND2:=IND2+1
     END;
    IND:=IND+1;
   END
END;(*REMPLACEMENT*)
```

```
PROCEDURE INSERTION (DIMFREQ,FREQNUL:INTEGER;VAR FREQUENCE:TAB7;
                     VAR ENTREE:CHAINE;VAR ORTHO:BOOLEAN);


VAR IND,IND2,COMPTLET,BIDON1,BIDON2,POSCRIT:INTEGER;
    LETGAU,LETDROI:CHAR;
    INS,TRAVAIL:CHAINE;
    LETINS:TAB26;
    POSDIGCRIT:TAB7;
    DIMPOSDIGCRIT:INTEGER;


 PROCEDURE CLASPOSINS (FREQNUL:INTEGER;VAR FREQUENCE,POSDIGCRIT:TAB7;
                       DIMFREQ:INTEGER;VAR DIMPOSDIGCRIT:INTEGER);

 VAR IND,IND2,MIN,PM:INTEGER;
 BEGIN
  IF FREQNUL <> 0
     THEN
       BEGIN
         IF FREQNUL = 1
            THEN POSDIGCRIT[1]:=1
            ELSE POSDIGCRIT[1]:=FREQNUL;
         DIMPOSDIGCRIT:=1
       END
     ELSE
       BEGIN
         FOR IND:=1 TO DIMFREQ DO
            BEGIN
              MIN:=MAXINT;
              FOR IND2:=1 TO DIMFREQ DO
                BEGIN
                  IF FREQUENCE [IND2] < MIN THEN
                     BEGIN
                       MIN := FREQUENCE[IND2];
                       PM:=IND2
                     END
                END;
              FREQUENCE [PM]:=MAXINT;
              POSDIGCRIT[IND]:=PM;
            END;
         DIMPOSDIGCRIT:=DIMFREQ
       END
 END;(*CLASPOSINS*)



 PROCEDURE LETGAUDROI (TRAVAIL:CHAINE;POSCRIT:INTEGER;VAR LETGAU,LETDROI:CHAR);

 BEGIN
   IF POSCRIT=1 THEN LETGAU:='@'
                ELSE LETGAU:=TRAVAIL[POSCRIT-1];
   IF POSCRIT>LENGTH(TRAVAIL) THEN LETDROI:='@'
                              ELSE LETDROI:=TRAVAIL[POSCRIT]
 END;
```

```
BEGIN
 CLASPOSINS(FREQNUL,FREQUENCE,POSDIGCRIT,DIMFREQ,DIMPOSDIGCRIT);
 IND:=1;
 TRAVAIL:=ENTREE;
 WHILE(IND<=DIMPOSDIGCRIT)AND(NOT ORTHO) DO
   BEGIN
     POSCRIT:=POSDIGCRIT[IND];
     LETGAUDROI(TRAVAIL,POSCRIT,LETGAU,LETDROI);
     CLASLETREMP(LETGAU,LETDROI,LETINS,COMPTLET);
     IND2:=1;
     WHILE (IND2<=COMPTLET)AND(NOT ORTHO) DO
       BEGIN
         INS:=LETINS[IND2];
         INSERT(INS,TRAVAIL,POSCRIT);
         RECHERCHE(TRAVAIL,ORTHO,BIDON1,BIDON2);
         IF NOT ORTHO THEN TRAVAIL:=ENTREE
                      ELSE ENTREE:=TRAVAIL;
         IND2:=IND2 + 1
       END;
     IND:=IND + 1
   END
END;(*INSERTION*)



PROCEDURE INTERVERTIR(COMMAX:INTEGER;VAR ENTREE:CHAINE;VAR ORTHO:BOOLEAN);

VAR INTER:CHAR;
    IND,BIDON1,BIDON2:INTEGER;
    STRLET,TRAVAIL:CHAINE;


BEGIN
 IF LENGTH(ENTREE)>=2 THEN
  BEGIN
   IND:=1;
   TRAVAIL:=ENTREE;
   WHILE (NOT ORTHO) AND (IND<=COMMAX+1) AND (IND < LENGTH(TRAVAIL)) DO
     BEGIN
       INTER:=TRAVAIL[IND+1];
       DELETE(TRAVAIL,IND+1,1);
       INSERT(CORSTR[INTER],TRAVAIL,IND);
       RECHERCHE(TRAVAIL,ORTHO,BIDON1,BIDON2);
       IF NOT ORTHO THEN TRAVAIL:=ENTREE
                    ELSE ENTREE:=TRAVAIL;
       IND:=IND+1
     END
  END
END;(*INTERVERTIR*)
```

```
PROCEDURE ESCAPE(POSDEPART:INTEGER;VAR ENTREE:CHAINE;
                 VAR COMMAX:INTEGER;VAR ORTHO:BOOLEAN);


VAR BI,BS,LONGTERM,DEBTERMENT,DEMI,INDCAR,DISTMIN,DISTCAND:INTEGER;
    TROUVE,CONTINUE:BOOLEAN;
    TERMENT,SUFFIXE,INTER:CHAINE;




  PROCEDURE DISTANCE(ENTREE,STR:CHAINE;DISTMIN:INTEGER;VAR DIST:INTEGER);

  VAR LONG,IND,IND2:INTEGER;
      STRBIS:CHAINE;
      RESULTAT:BOOLEAN;


   PROCEDURE INTERCHANGER(VAR RESULTAT:BOOLEAN);
   BEGIN
    RESULTAT:=FALSE;
    IF ((IND2+1) <= LENGTH(STRBIS)) AND ((IND+1) <= LENGTH(ENTREE))
      THEN IF (ENTREE[IND]=STRBIS[IND2+1]) AND (ENTREE[IND+1]=STRBIS[IND2])
             THEN
              BEGIN
               DIST:=DIST+1;
               IND:=IND+2;
               IND2:=IND2+2;
               RESULTAT:=TRUE
              END
   END;(*INTERCHANGER*)


   PROCEDURE SUBSTITUER(VAR RESULTAT:BOOLEAN);
   BEGIN
    RESULTAT:=FALSE;
    IF((IND+1) <= LENGTH(ENTREE)) AND ((IND2+1) <= LENGTH(STRBIS))
      THEN IF ENTREE[(IND+1)]=STRBIS[(IND2+1)]
        THEN
          BEGIN
            DIST:=DIST+1;
            IND:=IND+2;
            IND2:=IND2+2;
            RESULTAT:=TRUE
          END
   END;(*SUBSTITUER*)


   PROCEDURE AJOUTER(VAR RESULTAT:BOOLEAN);
   BEGIN
    RESULTAT:=FALSE;
    IF (IND+1) <= LENGTH(ENTREE)
      THEN IF ENTREE[IND+1]=STRBIS[IND2]
        THEN
          BEGIN
            DIST:=DIST+1;
            LONG:=LONG+1;
            IND:=IND+2;
            IND2:=IND2+1;
            RESULTAT:=TRUE
          END
   END;(*AJOUTER*)
```

```
PROCEDURE OUBLIER(VAR RESULTAT:BOOLEAN);
BEGIN
  RESULTAT:=FALSE;
  IF (IND2+1) <= LENGTH(STRBIS)
    THEN IF ENTREE[IND]=STRBIS[IND2+1]
      THEN
        BEGIN
          DIST:=DIST+1;
          LONG:=LONG-1;
          IND:=IND+1;
          IND2:=IND2+2;
          RESULTAT:=TRUE
        END
END;(*OUBLIER*)


(* DEBUT PROCEDURE DISTANCE *)

BEGIN
 STRBIS:=STR;
 LONG:=LENGTH(STRBIS);
 DIST:=0;
 IND:=1;
 IND2:=1;
 WHILE (IND<=LENGTH(ENTREE)) AND (IND2<=LENGTH(STRBIS)) AND (DIST<DISTMIN) DO
  BEGIN
   IF ENTREE[IND]=STRBIS[IND2]
    THEN
     BEGIN
      IND:=IND+1;
      IND2:=IND2+1
     END
    ELSE
     BEGIN
      INTERCHANGER(RESULTAT);
      IF NOT RESULTAT
       THEN
        BEGIN
         SUBSTITUER(RESULTAT);
         IF NOT RESULTAT
          THEN
           BEGIN
            AJOUTER(RESULTAT);
            IF NOT RESULTAT
             THEN
              BEGIN
               OUBLIER(RESULTAT);
               IF NOT RESULTAT
                THEN
                 BEGIN
                  IF(IND=LENGTH(ENTREE))OR(IND2=LENGTH(STRBIS))
                      THEN DIST:=DIST+1
                      ELSE DIST:=DIST+2;
                  IND:=IND+1;
                  IND2:=IND2+1
                 END
              END
           END
        END
     END
  END;
 DIST:=DIST+ABS(LENGTH(ENTREE)-LONG)
END;(*DISTANCE*)
```

```
PROCEDURE LISTCANDIDAT(ENTREE:CHAINE;COMMAX,POSDEPART:INTEGER;
                       VAR DISTMIN:INTEGER;VAR INTER:CHAINE);

VAR IND2,MAXCOM,IND3,LONGUEUR:INTEGER;
    MEILLEUR,SINTER:CHAINE;


BEGIN
 MAXCOM:=COMMAX;
 IND2:=POSDEPART;
 WHILE(MAXCOM=COMMAX)AND(IND2>0)AND(DISTMIN>1) DO
   BEGIN
     IND3:=INDICE[IND2];
     LONGUEUR:=INDICE[IND2+1]-IND3-1;
     CHARBYTE(IND3,LONGUEUR,SINTER);
     COMPARE(ENTREE,SINTER,MAXCOM);
     IF COMMAX=MAXCOM THEN
       BEGIN
         IND2:=IND2-1;
         DISTANCE(ENTREE,SINTER,DISTMIN,DISTCAND);
         IF DISTCAND<DISTMIN THEN
                              BEGIN
                                MEILLEUR:=SINTER;
                                DISTMIN:=DISTCAND
                              END
       END
   END;
 IND2:=POSDEPART+1;
 MAXCOM:=COMMAX;
 WHILE(MAXCOM=COMMAX)AND(IND2<DIMMOT)AND(DISTMIN>1) DO
   BEGIN
     IND3:=INDICE[IND2];
     LONGUEUR:=INDICE[IND2+1]-IND3-1;
     CHARBYTE(IND3,LONGUEUR,SINTER);
     COMPARE(ENTREE,SINTER,MAXCOM);
     IF COMMAX=MAXCOM THEN
       BEGIN
         IND2:=IND2+1;
         DISTANCE(ENTREE,SINTER,DISTMIN,DISTCAND);
         IF DISTCAND<DISTMIN THEN
                              BEGIN
                                MEILLEUR:=SINTER;
                                DISTMIN:=DISTCAND
                              END
       END
   END;
   INTER:=MEILLEUR
 END;(*LISTCANDIDAT*)
```

```
;  PROCEDURE LISTSUFFCANDI(SUFFIXE:CHAINE;POSDEPART:INTEGER;VAR DISTMIN:INTEGER
                           VAR INTER:CHAINE);
VAR IND2,IND3,LONGMOT,DEBTERM:INTEGER;
    MOT,SUFFCANDI:CHAINE;
    CH:CHAR;
BEGIN
  IND2:=POSDEPART;
  SUFFCANDI:=SUFFIXE;
  WHILE(SUFFCANDI=SUFFIXE)AND(IND2>0)AND(DISTMIN>1) DO
    BEGIN
      IND3:=INDICETERM[IND2];
      CHARBYTE(IND3,LONGTERM,SUFFCANDI);
      IF SUFFCANDI=SUFFIXE THEN
        BEGIN
          IND2:=IND2-1;
          CH:=LISTMOT[IND3];
          LONGMOT:=2;
          WHILE (CH<>'@')AND(IND3>0)DO
            BEGIN
              IND3:=IND3-1;
              LONGMOT:=LONGMOT+1;
              CH:=LISTMOT[IND3]
            END;
          IND3:=IND3+1;
          CHARBYTE(IND3,LONGMOT,MOT);
          DISTANCE(ENTREE,MOT,DISTMIN,DISTCAND);
          IF DISTCAND<DISTMIN THEN
            BEGIN
              INTER:=MOT;
              DISTMIN:=DISTCAND
            END
        END
    END;
  IND2:=POSDEPART+1;
  SUFFCANDI:=SUFFIXE;
  WHILE(SUFFCANDI=SUFFIXE)AND(IND2<=DIMTERM)AND(DISTMIN>1) DO
    BEGIN
      IND3:=INDICETERM[IND2];
      CHARBYTE(IND3,LONGTERM,SUFFCANDI);
      IF SUFFCANDI=SUFFIXE THEN
        BEGIN
          IND2:=IND2+1;
          CH:=LISTMOT[IND3];
          LONGMOT:=2;
          WHILE (CH<>'@')AND(IND3>0)DO
            BEGIN
              IND3:=IND3-1;
              LONGMOT:=LONGMOT+1;
              CH:=LISTMOT[IND3];
            END;
           IND3:=IND3+1;
           CHARBYTE(IND3,LONGMOT,MOT);
           DISTANCE(ENTREE,MOT,DISTMIN,DISTCAND);
           IF DISTCAND<DISTMIN THEN
             BEGIN
               INTER:=MOT;
               DISTMIN:=DISTCAND
             END
         END
      END
  END;(*SUFFCANDI*)
```

```
(* PROCEDURE ESCAPE  *)

BEGIN
  DISTMIN:=MAXINT;
  LONGTERM:=3;
  IF COMMAX>=4 THEN
                BEGIN
                   LISTCANDIDAT(ENTREE,COMMAX,POSDEPART,DISTMIN,INTER);
                   ORTHO:=TRUE;
                   ENTREE:=INTER
                END;
  IF DISTMIN=1
     THEN CONTINUE:=FALSE
     ELSE CONTINUE:=TRUE;
  BI:=1;
  BS:=DIMTERM;
  DEBTERMENT:=LENGTH(ENTREE)-2;
  TERMENT:=COPY(ENTREE,DEBTERMENT,3);
  WHILE CONTINUE AND(BI<=BS)DO
    BEGIN
      DEMI:=(BI+BS)DIV 2;
      INDCAR:=INDICETERM[DEMI];
      CHARBYTE(INDCAR,LONGTERM,SUFFIXE);
      IF TERMENT = SUFFIXE
        THEN
          BEGIN
            CONTINUE:=FALSE;
            LISTSUFFCANDI(TERMENT,DEMI,DISTMIN,INTER);
            ORTHO:=TRUE;
            ENTREE:=INTER;
          END
        ELSE IF TERMENT < SUFFIXE
          THEN BS:=DEMI-1
          ELSE BI:=DEMI+1
    END
END;
```

```
(* PROGRAMME PRINCIPAL   ................ *)


BEGIN
 CHARGEMENT;
 CONSMATDIG;
 CORSTRING;
 IND:=0;
 WRITE ('VOULEZ-VOUS COMMENCER ? (Y/N) : ');
 READLN (REPUTIL);
 WHILE REPUTIL <> 'N' DO
  BEGIN
   IND:=IND+1;
   WRITE ('MOT ',IND,' : ');
   READLN(ENTREE);
   RECHERCHE (ENTREE,ORTHO,BI,BS);
   IF ORTHO
    THEN WRITELN('ORTHOGRAPHE CORRECTE')
    ELSE
     BEGIN
      CONTCARCOM(BI,BS,ENTREE,COMMAX,POSDEPART);
      IF LENGTH(ENTREE)>6
       THEN ESCAPE(POSDEPART,ENTREE,COMMAX,ORTHO)
       ELSE
        BEGIN
         INTERVERTIR(COMMAX,ENTREE,ORTHO);
         IF NOT ORTHO THEN
          BEGIN
           CALPRODIGMOT(ENTREE,FREQUENCE,PROD,DIMFREQ,DIMPROD,FREQNUL);
           RECHMINPROD(PROD,DIMPROD,FREQNUL,POSLETCRIT,DIMPOSLETCRIT);
           SUPPRESSION(POSLETCRIT,DIMPOSLETCRIT,ENTREE,ORTHO);
           IF NOT ORTHO THEN
            BEGIN
             REMPLACEMENT(POSLETCRIT,DIMPOSLETCRIT,ENTREE,ORTHO);
             IF NOT ORTHO
                THEN INSERTION(DIMFREQ,FREQNUL,FREQUENCE,ENTREE,ORTHO);
            END
          END;
        END;
      IF ORTHO
       THEN WRITELN('PROPOSITION DE CORRECTION:',ENTREE)
       ELSE WRITELN ('JE NE CONNAIS PAS CE MOT')
     END;
   WRITE ('VOULEZ-VOUS CONTINUER ?(O/N)');
   READLN(REPUTIL)
 END
END.
```

ANNEXE A.VII.

EXEMPLES DE PHRASES GENEREES

VOUS CUISEZ CES GROS FROMAGES

TOI,LUI ET MOI BATISSONS UNE MAISON BLANCHE

VOUS ECRIVIEZ A CETTE REINE

ELLES OU LUI ONT PARLE A MON POLICIER

J'ENTENDAIS CETTE CHANSON

SEPT PETITES VOLEUSES ,SA PAUVRE AMIE ET SA JEUNE ELEVE CUEILLAIENT CETTE GRANDE TULIPE

VOUS OU VOUS EUTES CHANTE UNE CHANSON

VOUS AVEZ AIME CETTE INSTITUTRICE

L' ENFANT AURA PUNI CETTE MARCHANDE

J'EUS LAVE HUIT VETEMENTS

JE MANGERAI UN REPAS

VOUS,ELLES ET MOI AVONS AIME MON ENFANT

TOI ET VOUS BOIREZ SEPT EAUX

CE PETIT GARDIEN EUT REPONDU A CES HUIT BRUYANTES GAMINES

ON EUT JOUE HABILEMENT

L' AMIE AVERTIRA SEPT FEMMES

MON AMI TRISTE AURA RENDU NEUF SAUCISSES

ELLE AVAIT CHANTE AU GROS MENUISIER

ELLES PUNIRONT DES MUSICIENS BRUYANTS

LES GAMINES ET CE BON PHARMACIEN ENTENDAIENT CETTE JOUEUSE GAIE

LEURS SIX MECHANTS GENDARMES ACHETERENT CETTE VIEILLE ASSIETTE

ON AVAIT DIVERTI SON ENFANT

MAMAN EUT ECOUTE LA PRISONNIERE

VERONIQUE,ELLES OU ELLES AVAIENT CUIT TROIS PETITES TARTINES

VOUS,NEUF PRINCES OU TOI EUTES PUNI CE PETIT GARDIEN

UN ACTEUR AURA CUIT UN PETIT GATEAU

ON RENDRA AUX DIX REINES

TU EUS ATTAQUE SEPT JOUEUSES

CES JOUEUSES ONT PUNI DES ACTEURS

UN ENFANT TRISTE CHANTAIT A CE GENDARME HABILE

EUX,EUX ET TOI DESSINIEZ TROIS GROS TELEPHONES

CE CLIENT EUT ECOUTE HUIT SAVANTS HABILES

J'AVAIS PARLE A CES DIX JEUNES ACTEURS

ON AURA PUNI VOTRE SAVANT

CE PETIT ROI AVAIT LU CES BELLES FABLES

ELLES ATTRAPERENT DIX GAMINES GAIES

UN GARDIEN EUT COMPRIS CETTE COMEDIENNE BRUYANTE

ON LIT A L' ENFANT

SYLVIE OU SEPT BETES CHANTEUSES ONT DIVERTI DES PHARMACIENS

CETTE JEUNE FEMME AURA RECITE CE POEME TRISTE

IL MONTRE LES VIOLETTES

TU EUS LU AUX INSTITUTRICES JEUNES

LES HISTOIRES OU DINGO DIVERTIRONT CE PATRON

LEUR MECHANTE PATRONNE A CUIT LES CINQ SAUCISSES

TROIS ACTRICES BRUYANTES ONT ENDORMI UNE FEMME MALADE

TU AURAS DEPENSE RAPIDEMENT

TU CUEILLES DES ANANAS BLANCS

CES INSTITUTRICES OU LEUR GARDIENNE REGARDERONT LES GRANDES ASSIETTES

TU BATISSAIS SA PETITE MAISON

UNE MUSICIENNE A MONTRE A DES BOULANGERES HABILES

NEUF INSTITUTRICES AVAIENT MANGE LA SAUCISSE

THOMAS LAVA UN VIEUX HABIT

ILS PARLAIENT A NOS NEUF GRANDES REINES

ILS ONT RENDU A CE POLICIER

ELLES,CECILE OU UNE VOLEUSE AVAIENT LU LEURS CINQ VIEILLES CHANSONS

LE CHANTEUR AURA DEPENSE RAPIDEMENT

UN MARCHAND EUT RECUEILLI LES GRANDES CHANTEUSES

LA MARCHANDE A REGARDE UN PETIT DESSERT

ON RENDRA AUX DIX REINES

UNE ENFANT TRISTE ET ELLES ETUDIAIENT DES GRAMMAIRES

TOI OU NOUS DIVERTIRONS CES NEUF DOCTEURS

ELLES VENDENT AUX REINES

TOI OU VOUS ATTAQUATES TON AMI

NATHALIE,VOUS OU VOUS MANGEATES CES PAINS

DAVID DIVERTIT MON INSTITUTRICE

NOUS DIVERTIRONS UN MARCHAND MALADE

ELLES DEROBERENT DES SALAMIS

IL A ATTRAPE LES GRIPPES

LE JEUNE MUSICIEN DISAIT A QUATRE MUSICIENNES

ILS COMPRENNENT LES JOUEURS RAPIDES

ELLES DEROBAIENT DES PETITES BLOUSES

NATHALIE,ELLES ET CES SEPT AMIES AURONT MANGE CETTE PETITE SAUCISSE

ON EUT PROMIS LEUR TRAIN

EUX,LA MECHANTE INSTITUTRICE OU LUI AVAIENT CHANTE CETTE FABLE

VOUS ET VOUS DEPENSIEZ AUSSITOT

CATHERINE,VOUS ET VOUS AVIEZ MENTI AU PATRON

TU DEROBAS LA SAUCISSE

UNE ENFANT ,CINQ DOCTEURS ET ALICE AURONT ATTAQUE CETTE PRINCESSE MALADE

TU MENTAIS AUX CHANTEUSES

CE MAGICIEN DEPENSAIT PEU

JE MONTRAI A LA JEUNE CLIENTE

ON ENTEND TA VOLEUSE HABILE

ILS ONT JOUE GENTIMENT

VOUS ET NOUS MONTRERONS A VOS PERES MALADES

NICOLE,ELLES ET EUX ECRIRONT A LEURS HUIT PETITES SAVANTES

ELLES AURONT RENDU LES SEPT BETES ARMOIRES

CETTE GRANDE ACTRICE A OFFERT CETTE ASSIETTE BRUNE

ELLES OU VOUS AUREZ BU CES EAUX

LES TROIS BOULANGERS OU ELLE ONT MONTRE UN HABIT JAUNE

VOS AMIES EURENT ENTENDU DES PATRONS

ON EUT CUIT DEUX PAINS

ELLES ECOUTERENT CETTE MECHANTE ELEVE

LE GAI TROUBADOUR AURA DIT SA BELLE HISTOIRE

ELLES,ELLES OU VOUS AVEZ DEMANDE A LA BELLE MAGICIENNE

UN CHANTEUR VENDIT A HUIT MAGICIENNES ASTUCIEUSES

TU AVAIS DONNE A CE GENTIL DOCTEUR

ELLE ET CES DEUX BOUCHERES CUISENT CE REPAS

TU CUEILLES LA VIOLETTE

TINTIN EUT GUERI UNE GARDIENNE ASTUCIEUSE

ELLE ET MOI AURONS RENDU LE JAMBON

ELLES,TOI OU VOUS AVIEZ ENDORMI CES AMIS

NOUS RENDIMES AUX SEPT BOUCHERES ASTUCIEUSES

NEUF CHANTEURS ASTUCIEUX REPONDAIENT A CE BON PRINCE

NOUS ATTENDRISSIONS UNE MERE

LA REINE ,ELLES OU ELLE DEMANDAIENT A UNE CLIENTE

NATHALIE OU LEURS BOULANGERS CUISIRENT TON FROMAGE

ELLES,ELLE ET VOTRE BELLE MUSICIENNE VENDRONT A UNE BELLE MAGICIENNE

MONIQUE ET ELLES AVAIENT AVERTI NOTRE VOLEUR

PAPA DONNERA UN BETE COSTUME

J'AURAI OFFERT A NOS REINES

LES TROIS PRINCES AURONT PARLE A UN HOMME MECHANT

CE JOUEUR ACHETAIT A UN BOULANGER

ON A ECRIT A VOS SIX PRINCES BLANCS