UNIVERSITE

PARIS 7

THESE pour le Diplôme

de DOCTORAT

Spécialité: Didactique des Disciplines

Présentée par: Jean-Baptiste LAGRANGE

SUJET de la THESE:

Des situations connues aux traitements sur des données codifiées.
Représentations mentales et processus d'acquisition dans les premiers apprentissages en informatique.

VOLUME ANNEXES

Annexes au chapitre 5

Annexes au chapitre 6

Annexes au chapitre 7

Annexes au chapitre 8

Annexes au chapitre 9

Annexes au chapitre 10

Annexes au chapitre 12

Annexes au chapitre 5

Le chapitre **5** décrit les caractéristiques générales des langages à disposition des élèves débutants, et les spécificités du traitement des chaînes de caractères, et examine la façon dont ces éléments du langage se présentent pour l'élève débutant (précurseurs, difficultés attendues...). Pour cela, nous nous appuyons sur les résultats des recherches présentées au chapitre **1**. Parmi les caractéristiques des langages, nous nous intéressons particulièrement à l'itération en tant qu'action complexe de façon à préparer une étude expérimentale, qui sera rapportée au chapitre **6**. Le paragraphe **1.4** du chapitre **5** (les actions complexes; l'itération à un point de sortie) fait référence à deux types de travaux:

- les conceptions développées par Jacques Arsac dans ses ouvrages pour l'enseignement de l'informatique [ARSAC 83] et [ARSAC 80].
- deux études expérimentales, c'est-à-dire s'appuyant sur l'observation d'élèves en situation de résolution de problèmes [LABORDE, BALACHEFF, MEIJAS 1985] et [ROGALSKI 84].

De façon à ne pas alourdir le texte du chapitre **5**, la présentation et l'analyse de ces travaux sont dans cette annexe (paragraphes **1** à **3**) et les conclusions sont reprises dans le texte du chapitre au paragraphe 1.4. Nous avons placé également dans cette annexe une présentation des langages de programmation, et donné un rappel résumé de cette présentation dans le texte du chapitre au paragraphe **1.4**.

1. La structure itérative dans les ouvrages d'enseignement de la programmation de J.Arsac: présentation et discussion

L'apport de Jacques Arsac à la mise en place d'enseignements de l'informatique, ainsi que les objectifs de son ouvrage: "**Les bases de la programmation**" ([ARSAC 83]), ont été présentés au chapitre **2**. Nous avons tenté, dans ce chapitre, de confronter les méthodes pour l'écriture de programmes exposées par Jacques Arsac à d'autres types d'exposés. Nous présentons ici ce qui a trait plus particulièrement à l'itération; les conceptions de Jacques Arsac concernant l'itération influençant fortement l'enseignement de l'informatique au niveau où nous situons nos observations (option informatique des lycées), il importe de les présenter et de les discuter.

L'invariant d'itération: actions et situations

Le premier chapitre de "**Les bases de la programmation**" ([ARSAC 83]) est consacré à l'itération. La notion qui donne son sens à l'itération, est la notion d'"*invariant d'itération*", qui est elle-même un cas particulier d'"*assertion*": nous avons présenté ces notions au paragraphe **1** du chapitre **5**. Selon Jacques Arsac, l'idée d'assertion a été proposée comme un moyen de donner un sens à un programme. Il montre comment l'invariant d'itération permet d'interpréter, de construire, de corriger et de faire évoluer des programmes itératifs. L'accent mis sur les assertions en général, et sur l'invariant d'itération dans le cadre de l'itération est à mettre en relation avec l'hypothèse de Jacques Arsac sur les difficultés en informatique: le programmeur pense trop en termes d'actions («*Que va t'on faire* ?») alors que seules les situations («*Où en est-t'on maintenant* ?») donnent un sens aux programmes: «*Il*

faut absolument penser d'abord en termes de situations, les actions à faire se déduisant des situations».

L'invariant d'itération dans les premiers apprentissages

"**Les bases de la programmation**" est un ouvrage de niveau licence-maîtrise; la compréhension des concepts abordés suppose en effet une première expérience de la programmation; il est donc intéressant d'étudier comment le concept d'"invariant d'itération" est présenté et utilisé dans un ouvrage d'initiation: "**Premières leçons de programmation**" ([ARSAC 80]), bien que ne pouvant être considéré comme un manuel scolaire est un ouvrage explicitement destiné à des débutants; son l'influence sur l'enseignement est importante, particulièrement sur l'option informatique des lycées. Le domaine qui nous intéresse fait l'objet de deux chapitres. Le chapitre 3: "**La répétition**" concerne les itérations où l'on «*à répéter la même action pour tous les éléments d'une suite (...) La forme schématique en est: POUR TOUS les éléments d'une suite bien définie FAIRE une certaine action* ». Dans ce type d'itération, le nombre d'itérations est connu à l'avance, et il semble que pour l'auteur, l'action à répéter (c'est-à-dire le corps d'itération) suffit à donner son sens à l'itération; en effet il ne présente pas l'invariant d'itération dans ce chapitre. Pourtant, des structures dont l'interprétation par l'élève nous semble difficile, sont présentées dans le cadre de la répétition: comptage, accumulation, répétitions imbriquées.

Le chapitre 4: "**L'itération**" introduit "*l'invariant d'itération*" comme moyen de création d'une itération, sous le nom de «*hypothèse de récurrence*»; une méthode est présentée; elle concerne les problèmes pour lesquels l'action à répéter n'a pas de caractère d'évidence. Selon l'auteur, cette méthode comprend 5 phases, «*qu'il est vivement recommandé de prendre dans l'ordre*»:

- «*faire une hypothèse de récurrence* », (c'est-à-dire déterminer a-priori une situation invariante)
- «*voir si c'est fini* », (c'est-à-dire déduire une condition d'arrêt de la situation invariante)
- «*se rapprocher de la solution* » (c'est-à-dire construire les actions du corps de boucle),
- «*démarrer le calcul* » (fixer les valeurs initiales des variables)
- «*peigner le programme.*»

Discussion:

Y a t'il pertinence à distinguer "répétition" et "itération" ?

Dans la répétition, la condition d'arrêt est moins générale (puisqu'elle consiste à vérifier qu'une liste donnée a été parcourue), et elle n'interagit pas avec les actions du corps de boucle; en ce sens, une partie des difficultés que peuvent rencontrer les élèves ne sont pas présentes dans la répétition. Néanmoins, il ne nous apparaît pas qu'il existe un saut qualitatif qui justifierait une méthode d'analyse spécifique pour l'itération à un point de sortie générale; en effet, d'autres difficultés peuvent intervenir. Considérons, par exemple, d'une part un calcul cumulatif sur une suite déterminée (par exemple l'addition des N premiers entiers, où le "retournement" d'une chaîne de caractères) et d'autre part un problème de recherche d'une première occurrence. Le premier calcul est une répétition, mais entraîne d'une part la présence d'une variable d'accumulation, plus difficilement considérée qu'une variable compteur (voir [SAMURCAY 85]) et d'autre part la coordination entre cette variable d'accumulation et la variable compteur. Le second calcul est une itération plus générale,

mais une variable compteur suffit: il n'y a pas de coordination entre variables itératives, et l'initialisation et la condition d'arrêt s'en trouvent simplifiés.

La méthode exposée par Jacques Arsac pour l'itération est-elle de nature à répondre aux difficultés des élèves?

Il est indéniable que l'invariant d'itération rend des services en particulier pour la preuve des itérations et l'initialisation. Cependant, nous ne voyons pas de façon claire l'intervention des assertions au niveau de la construction de l'itération, et le schéma en cinq points de [ARSAC 80] nous semble hypothétique en tant que méthode de construction pour des débutants. En effet, les difficultés auxquelles les élèves seraient confrontés lors de la recherche d'une situation invariante à partir du seul texte d'un problème, et lors de la construction des éléments de l'itération à partir de cette situation invariante, paraissent d'un niveau conceptuel élevé: en effet, lors de la mise en évidence d'un invariant d'itération, la question n'est pas «*Où en est-on maintenant?*», mais plutôt «*Où en serions nous si le programme était construit?* »; il s'agit d'une démarche inductive analogue à la résolution de problèmes de construction en géométrie[1].. De même, la détermination des actions du corps de l'itération à partir de l'invariant d'itération consiste à chercher un procédé de passage d'une propriété `P(n)` à la propriété `P(n+1)`.; elle paraît donc d'un niveau conceptuel comparable au raisonnement par récurrence, et donc dépasser nettement celui d'un public hétérogène de débutants[2].

De la lecture du texte de R.Samurçay étudiée au chapitre **1** [SAMURCAY 85], et des deux textes qui suivent (paragraphes **2** et **3** de cette annexe), nous déduisons que, concernant les difficultés des débutants, de nombreux éléments interagissent: conception naïve du langage de programmation (qui conduit les élèves à calquer l'usage des avertisseurs d'itération sur leur équivalent dans le langage courant), absence de maîtrise de la logique propositionnelle (qui serait nécessaire en particulier pour l'écriture de la condition de sortie), difficulté d'articulation entre les différents éléments de l'itération (qui traduit la difficulté à prendre en compte les ruptures de séquentialité, et donc la temporalité de l'exécution). Les difficultés concernent donc aussi bien la compréhension des actions et de leur articulation que la conception logique de l'itération; elles ne se réduisent donc pas à l'absence de prise en compte d'une situation invariante. Avant de pouvoir se poser la question «*Où en est-t'on maintenant ?*», l'élève aurait à se poser la question «*Quelle est la signification des actions que je peux introduire dans le programme?* ».

Dans les cas les plus simples, la construction d'une itération peut s'envisager comme la découverte, dans un calcul, d'une *régularité* permettant son organisation en répétition; ce type de construction met l'accent sur les actions, l'invariant d'itération pouvant intervenir .pour les parties les plus délicates: preuve, initialisation, transformation du programme... Il nous semble, par analogie, que les élèves résolvent les problèmes de construction de figure en géométrie d'abord de façon naïve (au moins

---

[1]Par exemple, pour résoudre le problème: *construire, s'il existe, le centre d'un cercle passant par trois points donnés*, on supposera que ce cercle existe, on en déduira qu'il appartient nécessairement aux médiatrices des segment joingant deux des points donnés, ce qui suggère une construction. Réciproquement, cette construction aboutit si et seulement si les médiatrices ne sont pas parallèles, donc si et seulement si les trois points ne sont pas alignés.

[2]Les deux démarches supposent un processus de pensée que [ROGALSKI 91] qualifie de «*raisonnement sous hypothèse*».

jusqu'en seconde), utilisent des éléments logiques dans des situations de validation, avant de mettre en oeuvre la démarche articulant induction et déduction que nous avons évoquée ci-dessus. Notre hypothèse est donc la suivante: dans la construction d'une itération s'appuyant sur la découverte, dans un calcul, d'une *régularité* , les élèves débutants rencontrent et résolvent des difficultés, ce qui leur permet une première compréhension du schéma itératif; l'invariant d'itération est utilisé d'abord a posteriori pour les points délicats.

2. Problématique et genèse du concept d'itération, une approche expérimentale [LABORDE, BALACHEFF, MEIJAS 1985]

L'itération à un point de sortie

La forme générale de l'itération envisagée ici est `ITERER <C1> <condition> <C2> FIN ITERER` où `<C1>` et `<C2>` désignent deux séquences d'instructions, et `<condition>` une expression booléenne impliquant la sortie de la boucle. Il s'agit donc de ce que nous avons appelé l'itération à un seul point de sortie. Pour les auteurs, les caractéristiques de cette forme sont les suivantes

- double relation entre la condition et le corps de l'itération: «*la condition permet de contrôler la répétition du corps de l'itération, mais par ailleurs, celui-ci doit permettre la mise à jour des valeurs des variables de la condition*».
- nécessité d'une initialisation, qui «*ne concerne pas seulement les variables de la condition mais aussi éventuellement certaines variables apparaissant dans le traitement de l'information dans le corps de l'itération*».
- nécessité d'«*un avertisseur d'itération et (des) moyens de délimiter le corps de l'itération*».

L'étude expérimentale

Elle consiste à demander à des élèves l'écriture d'un algorithme destiné à être exécuté par un dispositif; les élèves sont des collégiens (14-15 ans) n'ayant reçu aucun enseignement d'informatique. Ce dispositif se trouve ici être un «*robot virtuel*» supposé exécuter des instructions suivant un code donné. Le robot est virtuel en ce sens qu'il n'y a pas de dispositif matériel réalisant les programmes: un expérimentateur simule l'exécution du programme, et commente le fonctionnement du robot. Les capacités supposées du robot sont les suivantes: tirer au hasard une boule dans une urne, lire un nombre inscrit sur cette boule, faire des additions. Le langage comporte des instructions ayant trait à ces capacités:

`PUB`: prendre une boule dans une boîte,

`LIS`: lire le nombre tracé sur la boule qu'il vient de prendre,

`COMPT`: le robot a un compteur qui augmente de 1 à chaque fois qu'on appuie sur la touche `COMPT`,

`TAPE`: il tape sur la calculette le nombre qu'il vient de lire,

`TAPE+`: il tape sur la touche +,

`TAPE=`: il tape sur la touche =

des instructions de contrôle parmi lesquelles

MARCHE, ARRET,

trois couples synonymes de d'avertisseurs d'itération

(REC,FINREC); (REP,FINREP);(CONT,FINCONT),

et deux instructions conditionnelles conduisant à l'arrêt de l'exécution, l'évaluation portant dans un cas sur le contenu de la boîte, dans l'autre sur la valeur du compteur.

La tâche des élèves est la suivante:

- dans une des modalités il s'agit d'écrire un programme pour que le robot retire de la boite un nombre donné de boules, et rende la somme des nombres inscrits sur ces boules.
- dans l'autre modalité il s'agit d'écrire un programme pour que le robot retire les boules tant qu'il y en a dans la boite et rende de même la somme des nombres inscrits sur ces boules.

Les conclusions

De l'observation et des conclusions, se dégagent notamment les points suivants:

- «*la signification et l'expression de l'itération évoluent*» au cours des étapes marquant la construction de l'itération. Le programme est d'abord une simple juxtaposition d'instructions indiquant que l'aspect répétitif des actions du robot n'a pas été perçu. Puis les avertisseurs d'itération sont utilisés, mais le sens que les élèves leur attribuent, et par conséquent, la façon dont ils les placent dans le programme n'est pas cohérent avec le langage (avertisseur de début d'itération placé après le corps d'actions à répéter; confusion entre l'avertisseur de fin d'itération et l'instruction de sortie de l'itération.) Ayant dépassé cette étape , les élèves doivent construire la «*sortie de la boucle*» c'est-à-dire concevoir et placer l'instruction alternative de sortie de l'itération.
- des difficultés apparaissent avec les instructions qui «*nécessitent une articulation avec d'autres instructions ou une coordination entre les composantes internes de l'instruction*». C'est particulièrement le cas de l'alternative de sortie d'itération: la partie condition dépend de la place que cette alternative occupe dans le corps de l'itération.
- des représentations erronées du fonctionnement du dispositif interviennent. Les représentations des élèves concernant le fonctionnement du robot peuvent être erronées, non seulement dans les anticipations servant à l'écriture du programme, mais aussi lors des simulations du fonctionnement du programme qu'ils effectuent dans l'intention de le valider. Ces représentations sont influencées par l'expression d'instructions itératives en langage naturel. Seule la simulation effectuée par un expérimentateur place les élèves devant l'effet véritable de leur programme. C'est particulièrement le cas de l'instruction de sortie d'itération; la nécessité de vérifier la condition de sortie à chaque pas de l'itération n'est pas ressentie spontanément.

3. Sur une séquence didactique en informatique [ROGALSKI 84]

L'étude porte sur l'alphabétisation informatique, c'est-à-dire «*les toutes premières heures (15 à 20) dans l'introduction à la programmation sur micro-ordinateur*», et en particulier sur la «*structure de boucle avec contrôle*». Elle comporte trois phases.

Première phase: explicitation d'une procédure répétitive simple.; étude des représentations initiales des élèves

La tâche consiste dans l'écriture d'un message «*à un opérateur disposant d'une machine* `A` *qui additionne, d'une machine* `B` *qui ajoute +1, pour que l'opérateur effectue le produit* `a x b`*, avec des données* `a et b` *entiers quelconques*». Les difficultés suivantes sont repérées:

- difficulté à «*passer d'une représentation des états à l'explicitation des transformations* » (ici il s'agit de passer d'une représentation statique de la multiplication comme addition multiple portant sur le même terme, à l'itération d'une action: ajouter le terme au résultat précédent).
- difficulté à «*acquérir une représentation séquentialisée de ces transformations*».
- difficulté à expliciter l'initialisation de la situation.

Les résultats concernant les représentations spontanées de la répétition, sont cohérents avec ceux de l'étude précédente:

- La description de l'action précède l'expression de la répétition, puis le test qui arrête cette répétition.
- «*l'opération de transformation des données (...) est représentée avant l'opération qui en contrôle le nombre d'exécutions*» (ici, l'addition de la donnée au résultat partiel précède l'incrémentation du compteur, ce qui n'a pas de conséquence, mais dans le cas général, le compteur intervient dans la transformation des données, et la place de l'incrémentation devient critique).

La deuxième phase: résolution d'un problème

Il s'agit lors de la 6ème séance de la résolution du problème suivant: «*écrire (sur machine en* **Pascal***) le programme qui calcule et affiche la somme des n premiers nombres entiers, n étant choisi par l'opérateur*». Cette phase permet d'établir des «*niveaux hierarchisés de difficulté dans le passage de l'expression et du calcul effectif à l'écriture de la boucle du programme demandé*». Les 3 niveaux suivants sont repérés:

- un groupe «*arrive au résultat en trois étapes plus une intervention finale de l'enseignant qui porte sur l'ordre interne au corps de boucle*».
  - Les 2 premières étapes consistent en une résolution à la main dans le cas de 5 nombres; d'abord (première étape) sous une forme (addition de deux en deux) difficilement transformable en itération, puis (2ème étape), sous la forme d'une sommation cumulative.
  - La troisième étape est l'écriture d'un programme **Pascal** utilisant la forme `REPETER...JUSQUA` et comportant une erreur dans l'ordre des instructions du corps de boucle. Dans cette forme, l'avertisseur de fin d'itération et le test d'arrêt sont confondus dans la même instruction, ce qui explique que la phase d'expression du corps de boucle, séparée de la conception du test d'arrêt, présente dans l'étude précédente, ne se rencontre pas ici.
- le protocole d'un second groupe «*témoigne de l'existence de plusieurs obstacles non franchis*»:
  - difficulté à utiliser un même nom de variable pour des valeurs successives différentes (c'est nécessaire pour le compteur d'itération, et pour la somme intermédiaire),

- impossibilité de dégager des invariants dans les étapes successives.

- un troisième groupe «*montre une absence d'assimilation*»; les difficultés apparues dans la phase précédente conduisent à un échec.

La troisième phase: étude des réponses individuelles des élèves à un questionnaire écrit

Cette phase se situe à la fin de la période d'alphabétisation.; elle a pour but l'étude de «*la fonctionnalité des variables dans les structures répétitives*». Les résultats sont cohérents avec [Samurçay 85]: si le compteur est compris «*comme élément intervenant à la fois dans le corps de boucle et dans le test*», il n'en est pas de même d'autres variables (résultat intermédiaire, donnée du problème). Par ailleurs l'initialisation, quand elle n'est pas réglée par une opération de lecture «*reste difficile à cette étape de l'initiation pour une partie importante des élèves*».

4.. Les structures générales et les fonctions sur les chaînes de caractères dans différents langages de programmation

Le travail d'analyse préparatoire constituant le chapitre **5** ne dépend pas d'un langage de programmation réellement existant; en effet, cette analyse et le travail avec les élèves rapportés dans les chapitres suivants doivent dépendre des notions informatiques visées, et non d'un langage de programmation: toute personne désirant reproduire la démarche décrite doit pouvoir le faire en utilisant le langage de programmation de son choix parmi une classe assez large. Pour notre part, en situation d'intervenir dans des classes dont nous n'avons pas la responsabilité, nous sommes dépendant du choix de l'enseignant, qui peut lui-même dépendre de contingences extérieures à notre propos (langage choisi dans l'établissement...). Il est important de noter que, avec des débutants, le langage d'expression des algorithmes est nécessairement très proche du langage de programmation utilisé; c'est pourquoi, dans les observations rapportées aux chapitres suivants, les énoncés que nous proposons aux élèves font référence au langage de programmation qu'ils utilisent. Les considérations qui suivent doivent montrer comment il est possible d'adapter ces énoncés à des classes utilisant un autre langage de programmation.

Nous commençons par décrire les caractéristiques de trois langages où les structures et fonctions présentées au chapitre **5** se retrouvent largement: il s'agit de **BASIC** (**§4.1**), **LSE** (**§4.2**)et **Pascal** (**§4.3**)[3]. Puis nous présentons des langages plus spécifiques: il s'agit de divers langages algorithmiques ou d'outils généraux (Tableurs, S.G.B.D.) (**§4.4**). Enfin nous présentons un langage (**LOGO**), lui aussi utilisé pour l'initiation, mais présentant des structures et fonctions différentes (**§4.5**). Ceci doit nous permettre de préciser les conséquences, pour les élèves, du choix d'un contexte de programmation *impératif*, et de la structuration des données à l'aide des *chaînes de caractères*. Nous ne prétendons évidemment pas donner une présentation exhaustive de chacun de ces langages. Nous donnerons des exemples de programme et un ensemble de sources bibliographiques.

---

[3]Ces trois langages sont admis pour la rédaction des programmes à l'épreuve optionnelle du baccalauréat faisant suite à l'option informatique. (note de service n° 87-304 du 1er octobre 1987)

4.1 BASIC

Ce langage est disponible sur toutes les machines accessibles pour l'initiation à l'informatique, et constitue un choix assez courant en initiation.

Sources bibliographiques

La première spécification du langage a été donnée par ses créateurs: [GARLAND 73][4]. A défaut de norme du langage, on peut se référer aux manuels d'utilisation des nombreuses versions, par exemple GWBASIC (Microsoft) disponible sur compatibles PC. Pour connaître l'"histoire" de **BASIC**, et des ses multiples évolutions, on peut lire [GATES 89][5]

Structures générales

Elles dérivent de l'organisation du programme en lignes numérotées. Les ruptures de séquentialité sont effectuées par *branchement* (éventuellement conditionnel) à une ligne donnée par son numéro (instruction **GOTO**). Les versions actuelles disposent de *l'alternative* complète (**IF** <condition> **THEN** <action1> **ELSE** <action2>), mais l'absence de parenthésage de bloc impose de gérer les alternatives en chaîne à l'aide de branchements (voir le style *GOTO* dans [GREEN 77] présenté au chapitre 1 de cette thèse). *L'affectation* est de la forme **LET** <identificateur> = <expression>, le mot **LET** étant généralement optionnel.

L'*itération* se construit à l'aide de branchements (voir l'exemple ci-dessous), et par conséquent, des régles strictes d'écriture doivent être observées si l'on souhaite que le programme reste lisible; la régle "*toutes les lignes sur lesquelles portent une instruction de branchement sont des lignes de commentaire*" constitue le minimum indispensable. **BASIC** dispose d'une seule forme d'itération "*structurée*" (c'est-à-dire non construite avec des branchements), la boucle **FOR** compteur = a **TO** b **NEXT**, où la variable numérique compteur prend successivement les valeurs de a à b. Un défaut courant des programmeurs **BASIC** est l'emploi de cette forme en dehors de son domaine de pertinence (on en verra un exemple au chapitre 6).

Fonctions sur les chaînes

Dans ses différentes versions, ce langage présente les fonctions du jeu restreint que nous avons présenté en **2.1.2** du chapitre **5** (longueur: **LEN**, concaténation: notée **+**, de façon infixée, sous-chaîne: **MID$**...), et des cas particuliers de la fonction sous-chaîne: **RIGHT$**(S$, n) pour sous-chaîne(S$, longueur(S$) - n + 1,n); **LEFT$**(S$,n) pour sous-chaîne(S$,1,n). La fonction position est présente, avec pour identificateur **INSTR**. Conformément aux principes du langage, il n'y a pas de déclaration de type des variables chaînes; une variable de type chaîne est caractérisée par l'adjonction d'un caractère **$** à la fin de l'identificateur. En cohérence, les identificateurs des fonctions dont le résultat est une chaîne, se terminent par un caractère **$**.

La réalisation peut différer d'une version à l'autre: gestion statique, où le nombre de caractères par chaîne est limité (en général à 255), et l'espace occupé par les chaînes est défini de façon constante, ou gestion dynamique qui ne présente pas ces limitations. Pour les exercices que nous proposerons, la gestion statique des chaînes ne

---

[4] **S.J.Garland** «BASIC a specification» *Dartmouth College HANOVER New Hampshire*

5 **B. GATES** «The 25th Birthday of BASIC» paru dans la revue *Byte* (octobre 89).

constitue pas un obstacle, et les élèves n'auront pas conscience des limitations qu'elle impose. Pour des problèmes réels, la gestion dynamique pourra être nécessaire.

Exemple de programmation

traduction en **BASIC** de l'algorithme de recherche de la `position` d'une chaîne comme sous-chaîne d'une chaîne donnée (énoncé au chapitre **5** paragraphe **2.1.4**).Les mots réservés du langage sont en caractères gras.

Entrée des données au clavier	`10 INPUT chaine1$, chaine2$`

Initialisation, calcul de longueur	`20 LET N = 0`
	`30 LET L1 = LEN (chaine1$)`
	`40 LET L2 = LEN (chaine2$)`

Cas où la recherche n'a pas d'objet	`50 IF L2 < L1 THEN GOTO 120`
Commentaire marquant le début de l'itération et incrémentation du compteur	`60 REM début itération` `70 LET N = N + 1`

Alternatives de sortie conditionnelle	`80 LET X = MID$ (chaine2$, N, L1)`
	`90 IF X = chaine1 THEN GOTO 120`
	`100 IF L2 < L1 + N THEN LET N = 0:GOTO 120`

bouclage	`110 GOTO 60`
Commentaire marquant la fin de l'itération.et affichage résultat	`120 REM fin de l'itération` `130 PRINT N`

4.2 LSE

Ce langage a été conçu en France, puis utilisé essentiellement dans l'enseignement secondaire pour l'enseignement de l'informatique et le développement de logiciels. Selon J. Hebenstreit (Préface à:[CANAL 1983]), il est «*utilisé par de nombreux enseignants du secondaire, mais aussi par un certain nombre d'Universités tant françaises que francophones*». Sa création coïncide avec le Colloque CERI/OCDE de Sèvres (Mars 1970) qui lance l'informatique dans l'enseignement.

Sources bibliographiques

Pour une présentation générale du langage et une initiation à la programmation:
[CANAL 1983] Parler LSE et apprendre à l'utiliser Eyrolles

Pour prendre connaissance de la norme (AFNOR)
[LSE 8086/8088]: manuel d'utilisation Microdur 1984

Pour prendre connaissance des conditions de la création et de l'évolution de **LSE**
[NOYELLES 89] Yves Noyelles La saga du **LSE** et et de sa famille (article paru dans la revue de l'EPI n°54 juin 89)

Structures générales

Comme dans **BASIC**, le programme est organisé en lignes numérotées. L'instruction de *branchement* similaire au `GOTO` de **BASIC** s'écrit **ALLER EN** <n° de ligne>. On dispose de *l'alternative complète* (`SI` <condition> **ALORS** <action1> **SINON** <action2>), avec un parenthésage de bloc (**DEBUT...FIN** limité à l'alternative) permettant d'imbriquer des alternatives (dans la mesure où elles tiennent dans la ligne). L'*affectation* s'écrit <identificateur>←<expression>.

L'*itération* peut se construire à l'aide de branchements comme en **BASIC**. Mais LSE dispose de formes d'itération "*structurées*" . La boucle **FAIRE** <n°de ligne> **POUR** compteur ← a **JUSQUA** b, où, comme dans la boucle **FOR...NEXT** de **BASIC**, la variable numérique `compteur` prend successivement les valeurs de a à b . Il existe également une forme `TANT QUE` (que nous avons présentée sur un plan général au chapitre 5 § 1.4. comme cas particulier d'itération à un point de sortie); elle s'exprime **FAIRE** <n°de ligne> **TANT QUE**<condition>, et le programmeur

peut "mixer" ces deux formes d'itération (voir ci-dessous l'exemple de programmation).

Fonctions sur les chaînes

En **LSE**, les chaînes et les booléens se *déclarent* par une instruction spéciale (voir ci-dessous exemple de programmation).

Les *fonctions* que nous avons proposé au **§ 2.1.2** du chapitre **5** comme jeu restreint suffisant à définir la structure, sont présentes: la concaténation se note de façon infixée (opérateur !), la fonction longueur se note `LGR` , la fonction `sous-chaîne` se note **sch.** Selon les promoteurs de **LSE**.«*Un des avantages du* **LSE** *est de posséder de nombreuses fonctions sur les chaînes de caractères, rendant la manipulation des dites chaînes très aisées.*» ([CANAL 83]). De fait, le jeu de fonctions est bien plus étendu que le jeu restreint que nous avons proposé ci-dessus [6]; Chaque fonction peut se construire à partir du jeu restreint. Ce jeu de fonction étendu s'est justifié à une époque pour l'écriture de logiciels: un traitement de texte (**TEXTE**), un assembleur-deboggeur, des programmes d'analyse de texte ont été programmés en **LSE**; mais ces logiciels n'ont pas connu d'utilisation en dehors du domaine des applications pédagogiques, et sont aujourd'hui très largement dépassés par des logiciels conçus à l'aide d'outils modernes de programmation utilisant des structures de données plus puissantes. Concernant l'apprentissage de la programmation, on peut s'interroger sur la validité de l'acquisition d'un jeu aussi étendu de fonctions; il nous semble plus intéressant de construire ces fonctions à l'aide du jeu restreint que nous avons indiqué, chaque fonction étant construite par les élèves comme élément de solution à un problème.

Les variables chaînes sont déclarées sans indication de longueur; la gestion des chaînes est dynamique. «*La longueur d'une chaîne n'est limitée que par la place disponible en mémoire*» [LSE 8086/8088]

Exemple de programmation

traduction en **LSE** de l'algorithme de recherche de la `position` d'une chaîne comme sous-chaîne d'une chaîne donnée (énoncé au chapitre **5** paragraphe **2.1.4**). Les mots réservés du langage sont en caractères gras.

---

[6] par exemple, la fonction SCH peut avoir comme troisième argument une chaîne et non un cardinal : dans ce cas, la sous-chaîne résultat s'arrêtera avant la première occurrence d'un caractère de la chaîne troisième argument ; ainsi cette chaîne constitue une liste de caractères d'arrêt. Si l'on souhaite par exemple, extraire le premier mot d'une phrase affectée à la variable phrase, on calculera SCH (phrase, 1, ' ,;:!'), de façon que l'extraction s'arrête avant le premier espace ou signe de ponctuation.
De plus la fonction SCH peut prendre un quatrième argument qui est obligatoirement une variable numérique (et non une expression) ; en effet, c'est un paramètre "*compte-rendu*" qui est modifié par l'appel de la fonction
De même, la fonction position existe avec plusieurs variantes (POS, SKP, PTR)

déclaration de type chaîne
déclaration de type booléen
entrée des données au clavier
calcul de longueur

booléen de contrôle de boucle
itération avec compteur et
condition de continuation

en ligne 100, astérisque (commentaire)
marquant la fin de l'itération.
à cause de la gestion du compteur par la
boucle **FAIRE**, le résultat est N-1

```
10 chaine c1, c2, x
20 booleen fini
30 lire c1, c2
40 L1 ← LGR (c1)
45 L2 ← LGR (c2)
50 fini ← (L2 < L1)
60 FAIRE 100 POUR N ← 1
                    TANT QUE NON fini
70     X ← sch (c2, N, L1)
80     fini ← (X=c1) OU (L2 < L1 + N)
90     SI fini ET NON (X=c1)
                    ALORS N ← 0
100 * fin de la boucle
110 afficher N-1
```

4.3 Pascal:

Sources bibliographiques

Description de **Pascal** standard par son créateur:
[JENSEN WIRTH 78]: K.Jensen N.Wirth (PASCAL User Manual and Report Springer Verlag 1978). Traduction française: Pascal, manuel de l'utilisateur Eyrolles 1980

Pour une version répandue de **Pascal**:
Manuel de TurboPascal Borland.

Structures générales

N. Wirth a utilisé **ALGOL** comme base de conception de **Pascal**, dans le but de «*disposer d'un langage adapté à l'enseignement de la programmation*». Dans ce but, le programme **Pascal** n'est pas organisé en lignes, et le recours aux branchements est systématiquement découragé. L'organisation du programme en *blocs* (de délimiteurs **BEGIN** et **END**) permet l'imbrication sans restriction des *alternatives*, conduisant au style décrit comme "*imbriqué*" dans [GREEN 77] présenté au chapitre **1** de cette thèse. Les marqueurs d'alternative sont **IF...THEN...ELSE**, comme en **BASIC**, et il existe une *alternative par cas* (CASE OF ...). L'*affectation* s'écrit <identificateur>:= <expression>.

L'*itération* ne se construit pas à l'aide de branchements comme en **BASIC** ou **LSE. Pascal** dispose au contraire de formes d'itération "*structurées*" plus éloignées de la représentation de l'exécution. La boucle **FOR** compteur:= a **JUSQUA** b **DO** <bloc actions> où, comme dans la boucle **FOR...NEXT** de **BASIC**, la variable compteur prend successivement les valeurs de a à b , mais n'est pas nécessairement numérique. Il existe également les formes TANT QUE et REPETER JUSQU'A (que nous avons présentées au chapitre **5** § 1.4. comme cas particulier d'itération à un point de sortie); elles s'écrivent respectivement **WHILE** <condition> **DO** <bloc actions>, et **REPEAT**<bloc actions> **UNTIL** <condition>[7]. La dernière

[7] [ARSAC 83].note comme un point faible de **Pascal** l'absence de forme d'itération permettant une expression aisée de l'itération à plusieurs points de sortie. Comme on le voit dans l'exemple de programmation ci-dessous, l'utilisation de WHILE ... DO pour coder une itération à deux points de sortie oblige à introduire artificiellement le booléen fini.

forme paraît la plus intuitive: <condition> est la condition qui doit être réalisée en sortie de boucle, et le bloc d'action n'a pas à être parenthésé. Nous l'utiliserons pour les observations rapportées aux chapitres **8** et **9**, non en **Pascal**, mais dans un **BASIC** "structuré" reprenant assez largement les structures générales de **Pascal**.

Fonctions sur les chaînes

En **Pascal**, les variables chaînes font l'objet d'une *déclaration* précisant le type, comme toute variable quelqu'en soit le type.

Pascal standard (c'est-à-dire conforme à [JENSEN WIRTH 78]) ne dispose pas des *fonctions* sur les chaînes de caractères: «*Les chaînes d'un seul caractère sont des constantes de type standard* `CHAR`; *celles qui contiennent* `n` *caractères* (`n>1`) *sont des constantes du type défini par*: `packed array [1..n] of CHAR` ». Ainsi, on peut, de façon primitive, accéder à un caractère connaissant son rang, ou le modifier, mais il n'y a ni concaténation, ni longueur, ni sous-chaîne. En pratique, on trouve des versions sur micro-ordinateur où une chaîne est un vecteur (tableau de dimension 1) dont l'élément d'indice 0 représente la longueur, les autres éléments étant les caractères; la longueur est donc limitée (en pratique à 255 caractères), et où l'on dispose des fonctions `concaténation`, `longueur`, et `sous-chaîne`, parmi d'autres. De plus, le type chaîne peut être résultat d'une fonction, ce qui n'est pas le cas des tableaux. Parmi les classes que nous avons observées dans le cadre de la préparation de cette thèse, deux utilisaient une version de **Pascal** de marque Borland (**TurboPascal**) disposant des fonctions **`copy`** `(sous-chaîne)`, **`length`** `(longueur)`, **`concat`** `(concaténation)`, la concaténation existant également sous forme infixée (opérateur +), et **`pos`** `(position)`. On trouvera au chapitre **7** quelques particularités découlant de ce choix.

Aperçu sur la réalisation d'une structure de chaîne

La non-disponibilité, dans **Pascal** standard de la structure de chaîne est l'occasion d'évoquer la réalisation d'une structure de chaîne conforme à ce que nous avons indiqué au chapitre **5**, cette réalisation étant une exemple de *représentation* au sens du chapitre **2**, et pouvant constituer un niveau d'explication possible. Dans l'exemple ci-dessous, nous donnons la réalisation des fonctions `longueur` et `sous-chaîne` dans le cas de chaînes représentées par un vecteur de caractères de taille fixe; on verra que dans cette réalisation, si une variable `X` prend comme valeur une sous-chaîne d'une variable `Y`, les caractères de `Y` composant `X` sont copiés (*dupliqués*) physiquement.

Si l'on souhaite réaliser une structure de chaîne dynamique (chaînes de longueur quelconque) on pourra utiliser les structures **Pascal** adaptées: une chaîne sera par exemple un *article* (**`record`**) composé d'un *entier* (**`integer`**) représentant la longueur de la chaîne, et d'un *pointeur* (**`pointer`**), représentant l'adresse d'un bloc mémoire contenant les caractères de la chaîne. Dans ce cas, une variable `X` ayant pour valeur une sous-chaîne d'une variable `Y` pourra être physiquement composée de caractères de `Y`: *il n'y a pas duplication*. Pour la réalisation des fonctions, dans le cas

---

Les conséquences de cette limitation n'apparaîssent pas dans la suite de cette thèse puisque nous nous limitons à l'itération à un point de sortie.

d'une gestion dynamique, on pourra consulter un ouvrage sur la compilation, par exemple [THALMAN LEVRAT 79][8]

Cet aperçu sur la réalisation de la structure de chaîne nous montre que des incertitudes sur la façon dont le dispositif traite les chaînes peuvent être fondées (nous verrons au chapitre 9 un exemple d'élèves rencontrant ces incertitudes): après l'affectation `X := Y` si les variables `X` et `Y` sont numériques, elle auront la même valeur, mais physiquement, cette valeur existera deux fois dans la mémoire. Par contre s'il s'agit de variables chaînes, dans une réalisation dynamique, après l'affectation `X = sous-chaîne (Y,n,p)`, `X` et `Y` peuvent avoir physiquement des caractères en commun, et il n'est pas évident de montrer qu'une modification de `X` n'a pas d'influence sur `Y`.

Exemple de programmation

Nous nous sommes placés dans le cas d'un **Pascal** standard, et nous avons défini les types et fonctions utiles (types: *chaine, ordinal, cardinal,* fonctions *longueur, sous-chaine, egal,* procédure *lire*). Le programme principal traduit l'algorithme de recherche de la `position` d'une chaîne comme sous-chaîne d'une autre énoncé au **2.1.4.** Ces types et fonctions spécifiques des chaînes sont en italiques, les mots réservés du langage sont en caractères gras.

les types liés à la structure de chaîne: `string[255]` a les propriétés de `array[0..255] of char`, et peut, de plus, être résultat d'une fonction

la longueur est la valeur du premier élément du tableau représentant la chaîne

le calcul d'une sous-chaine implique la copie des caractères concernés

```
program traitement_chaine;
type chaine = string[255];
     ordinal = 1..255;
     cardinal = 0..255;

var c1,c2,x: chaine;
    n,L1,L2:cardinal;
     fini:boolean;


function longueur(c:chaine):cardinal;
begin
    longueur:=ord(c[0])
end;{longueur}

function
souschaine(c:chaine;rang:ordinal;nbcar:cardinal):
                                              chaine;
var deplacement: cardinal;
begin
 for deplacement:= 0 to (nbcar - 1) do
  souschaine[1+deplacement]:=c[rang+deplacement];
 souschaine[0]:=char(nbcar)
end;{souschaine}
```

[8] **D.Thalman B.Levrat** «Conception et implantation de langages de programmation» *Gaëtan Morin* 1979

La fonction qui teste l'égalité de deux chaînes; l'égalité des éléments d'index 0 équivaut à l'égalité des longueurs; on teste tous les éléments jusqu'à en trouver deux qui différent ou la fin d'une chaîne.

```
function egal(c1,c2:chaine):boolean;
var i:0..255;
     egalaux:boolean;
begin
     egalaux:=true;i:=0;
     while egalaux and not(i=longueur(c1)+1) do
          begin
               egalaux:=(c1[i]=c2[i]);
               i:=i+1
          end;
     egal:=egalaux
end {egal };
```

La procédure permettant l'entrée d'une chaîne au clavier; l'utilisateur appuie sur la touche Retour pour signifier la fin de l'entrée, ce qui génère un CR+LF

```
procedure lire(var c:chaine);
const RC=char(13);
var r:char;i:cardinal;
begin
     read(r);
     i:=0;
     while not (r=RC) do
     begin
          i:=1+i;
          c[i]:=r;
          read(r)
     end;
     c[0]:=char(i);
     read(r);{pour lire le LF restant}
end {lire };
```

entrée des données au clavier
calcul de longueur
booléen de contrôle de boucle
initialisation
marqueur d'itération

```
begin {programme principal}
 lire(c1);lire(c2);
 L1:= longueur(c1);
 L2:= longueur(c2);
 fini:=(L2 < L1);
 n:=0;
 while not fini do
 begin
     n:=1+n;
     x:=souschaine(c2,n,L1);
     fini:= egal(x,c1) or (L2 < L1 + n);
     if fini and egal(x,c1) then n:=0
 end;
 writeln(n)
end.{programme principal}
```

fin du corps de l'itération.
affichage du résultat

4.4 D'autres langages impératifs possibles

Langages algorithmiques

Des langages algorithmiques (exécutables, ou non exécutables) ont été construits dans le but de fournir aux débutants un environnement de construction d'algorithmes et de programmation en rapport avec les conceptions des auteurs de ces langages concernant l'acquisition des premières structures. Le but de ce paragraphe n'est pas de discuter l'opportunité de tel ou tel langage algorithmique, mais de signaler ceux de ces langages qui, à notre connaissance, disposent des structures générales (impératives) et des fonctions sur les chaînes de caractères conformes à ce que nous avons présenté au chapitre **5**, et permettent donc, sans difficulté majeure, de transposer les observations décrites aux chapitres suivants; ajoutons que ces langages ne permettent pas le *branchement* et comportent donc une (des) forme(s) d'itération "*structurée*". Parmi les langages explicitement destinés aux débutants, on peut citer:

- **ALADIN**: source [RICHARD 85] C. et D. Richard Programmatique Belin 1985
- **MEDEE** : source [DUCRIN 84] A. Ducrin Programmation Dunod 1984
- **PEGASE**: source [HEUZE 90] F.Heuze Une démarche pour l'apprentissage de la programmation en classe de seconde (2nd colloque francophone sur la didactique de l'informatique 1990).

Tableurs, ou Systèmes de gestion de bases de données (SGBD)

Ces logiciels sont conçus en fonction d'une classe de problèmes impliquant une organisation particulière des données: données organisées en fichier dont les enregistrements sont constitués en champs dans le cas d'un SGBD, données organisées en tableau dans le cas d'un tableur. Certains choix pédagogiques peuvent conduire à l'utilisation de ces outils (ou plutôt de leur langage de commande) dans le cadre de l'initiation. Là aussi, il ne s'agit pas de discuter l'opportunité de l'emploi de tel ou tel de ces outils, mais de signaler qu'ils disposent des fonctions sur les chaînes de caractères conformes à ce que nous avons présenté au chapitre 5, et dans une certaine mesure, des structures générales (impératives).

Un S.G.B.D. comme **Dbase** (Aston Tate, distribué en France par La commande Electronique) offre la possibilité de champs de type "chaîne de caractères", et le langage d'interrogation et de programmation comporte le jeu réduit de fonctions que nous avons indiqué, en même temps que de structures de programmation impératives. De même les cellules d'un tableur comme **Multiplan**.(Microsoft) peuvent comporter des données de type chaîne, et le langage d'expression des *formules* comporte les fonctions du jeu réduit. Les structures de programmation sont par contre originales: une étude de l'acquisition par des débutants de ces structures dans le cadre du calcul algébrique, été menée par B.Capponi et N.Balacheff ([CAPPONI BALACHEFF 89])[9] et montre l'existence de difficultés spécifiques à la programmation sur un tableur conduisant à ce qu'il n'y ait pas transfert automatique des connaissances algébrique vers l'environnement du tableur. Il n'y a pas, à notre connaissance, d'étude concernant les chaînes de caractères dans ce domaine

4.5 LOGO:

A la suite des idées avancées dans [PAPERT 80] **LOGO** a été utilisé comme langage de programmation pour des activités pédagogiques, en particulier d'initiation à la programmation. Bien s'il ne s'agisse pas d'un langage autorisé pour l'épreuve optionnelle au baccalauréat, on trouve des compte-rendu d'utilisation en option informatique en classe de seconde (voir par exemple [BENETOLLO 87][10]). **LOGO** ne fait pas partie de la classe des langages *impératifs*; c'est pourquoi ses structures générales ne correspondent pas à ce que nous avons présenté au chapitre 5. Nous ne discuterons pas des différences qui peuvent exister entre les structures générales de langages *applicatifs* comme **LOGO**, ni des conséquences de ces différences au niveau cognitif. Le sujet de cette thèse étant en rapport avec l'acquisition de traitements sur des données de nature diverse, la structuration des données en **LOGO** (structure en *mots* et *listes*) étant originale par rapport aux données numériques, chaînes et booléens que nous

---

[9] **B. Capponi et N. Balacheff** «Tableur et Calcul Algébrique» in *Educational Studies in Mathématics* 21 1989.

[10] **R. Benetollo** «Place dans l'option informatique de langages tels que LOGO, PROLOG et LISP» in *Pratiques et Savoir-faire des élèves de l'option informatique*.Publié par le Ministère de l'Education nationale 1987.

envisageons dans cette thèse, il nous a semblé intéressant d'examiner ce que le choix d'une classe de langage implique sur la conception des données par les élèves.[11] C'est pourquoi, nous présentons brièvement les objets et fonctions représentatifs de données de type texte,présents dans ce langage,

Sources bibliographiques

[PAPERT 80] S. Papert "Mindstorm" Basic Books 1980 traduction française:"jaillissement de l'esprit" Flammarion,

[BENETOLLO 87] place dans l'enseignement de l'option de langages comme LOGO, PROLOG, LISP R. Benetollo in Pratiques et savoir faire)

[LOGO Manuel] D.Avram T.Savatier M.Weidenfeld LOGO-Manuel de Référence CEDIC-Nathan 1985

[BOURBION 84] M.Bourbion & alter L'alternative LOGO Armand Colin-Bourrelier.

La structuration des données en LOGO

Les fonctions sur des données textuelles (mots, listes)

Bien qu'il soit connu surtout pour l'univers de programmation que constitue la tortue graphique, LOGO dispose de structures permettant de traiter des données textuelles. Les objets de base sont les **MOTS**, assemblages de caractères de l'alphabet étendu; un nombre est le **MOT** formé des chiffres de sa représentation décimale; ainsi la différenciation existant dans les autres langages étudiés, entre un nombre et la suite des caractères de sa représentation externe n'existe pas ici; **LOGO** donne la priorité au caractère intuitif des données sur l'efficacité, ce qui n'est pas sans entraîner des incohérences: il n'est pas possible en effet de distinguer, par exemple le mot "00 du mot "0, ni "1000 de "1E4. La structure fondamentale est comme en **LISP** ou en **PROLOG** la `liste`; les `listes` de **LOGO** sont des listes symétriques (c'est à dire qu'un double chaînage permet d'accéder aux éléments d'une liste par la droite comme par la gauche) caractérisées par les fonctions:

```
MetPremier   (objet, liste) →liste (où objet désigne un mot
ou une liste).
PREMier      liste →objet
SaufPremier  liste →liste
```

avec les propriétés:

```
PREMier ( MetPremier:A:B ) =:A
SaufPremier ( MetPremier:A:B ) =:B
```

on a les fonctions symétriques, permises par le double chaînage: `MetDernier, DERnier , SaufDernier`

---

[11] De nombreux auteurs justifient leur préférence pour la programmation applicative (utilisation de **LOGO, LISP, SCHEME**) ou logique (utilisation de **PROLOG**) par la structure *algorithmique* du langage. (voir par exemple les nombreuses communications sur ce sujet au congrès francophone de didactique 1990). Le plus souvent, les conséquences sur le traitement des données ne sont pas prises en compte.

L'absence de typage des objets

Les mêmes fonctions opèrent sur les **MOTS**, considérés comme les listes de leurs caractères, ce qui permet que les fonctions opèrent indifféremment sur les nombres, les MOTS et sur les listes, et donc de *contourner le typage* des objets. Nous avons vu que ce choix peut conduire à des incohérences concernant les **MOTS** représentant des nombres. Signalons une autre de ces incohérences concernant les **listes** et les **MOTS**: la signification de la fonction **PREM** est de "faire descendre" dans les niveaux d'une liste, et donc l'application répétée de **PREM** sur une liste devrait se terminer par une erreur, la profondeur d'une liste ne pouvant être infinie: or, dès que la fonction **PREM** rend un **MOT**, l'application suivante donne le premier caractère de ce **MOT**, et la répétition boucle sur ce caractère.

Cette absence de typage des objets dans **LOGO** est à mettre en rapport avec la volonté des promoteurs de langage d'offrir un cadre non contraignant pour l'expression des "*idées spontanées*" de l'apprenti programmeur. Les incohérences que nous signalons ci-dessus montrent qu'on ne peut aller très loin dans cette direction. Il nous semble que le typage est une des contraintes spécifiques aux dispositifs informatiques; nous attendons pour notre part que les élèves rencontrent cette contrainte en connaissance de cause; après les observations que nous projetons, nous serons en mesure de dire si cette rencontre conduit à une meilleure représentation du dispositif.

Un exemple de parcours de liste se ramenant à l'itération

Ces fonctions d'accès ou de construction de liste, en cohérence avec la structure algorithmique du langage sont orientées vers le parcours *récursif* d'une liste. Mais dans les cas les plus simples, la récursion s'interpréte comme une itération, et nous allons montrer qu'un parcours itératif d'une chaîne de caractères peut se substituer au parcours de liste, le **fonctionnement mental** nécessaire au raisonnement étant cependant différent. Donnons un exemple assez courant (il s'agit d'établir une correspondance entre un nom et une note chiffrée de 0 à 20) et comparons une solution applicative (écrite en **LOGO**) et une solution impérative (écrite en **Pascal**).

Pour une solution applicative, on introduit une liste constante de doublets (`nom, note`) affectée à `"LISTENOMS`, puis un nom étant donné, on parcourt récursivement cette liste, jusqu'à trouver le doublet dont le premier élément est le nom cherché, et on rend le second élément; si le nom n'est pas trouvé, on rend `"inconnu"`. Le raisonnement consiste à repérer les deux cas où, compte-tenu des fonctions d'accès, la correspondance s'établit facilement (cas où la liste est vide, et cas où le premier doublet contient le nom cherché), puis, dans le cas général, à se rapprocher de ces cas particuliers, en considérant la liste privée de son premier doublet:

```
DONNE "LISTENOMS [[ARTHUR 6][EMILE 12][MARIE 16][FRED
6][JOEL 18][JANINE 7]]

POUR NOTE:N:L
   SI VIDE?:L [RENDS "inconnu]
   SI EGAL?:N PREM PREM:L [RENDS DER PREM:L]
   RENDS NOTE:N SP:L
FIN

ECRIS NOTE "FRED:LISTENOMS
```

Avec un langage impératif, on utilisera une chaîne de caractère constante où un caractère spécial ([) marque le début d'un nom. On fera parcourir la chaîne à l'aide d'un index `n` et, en présence du caractère spécial, on regardera si le nom cherché suit, et

dans ce cas, on calculera la note sous forme d'une sous-chaîne. Par exemple, en **Pascal** (Version **TurboPascal**)

```
program cherche_notes;

const
LISTENOMS:string[255]='[ARTHUR06[EMILE12[MARIE16[FRED06[JOEL
18[JANINE07';
var nom:string[10];
    note:string[10];
    n:1..255;
    l:1..10;
begin
 readln(nom);
 n:=1; L:=length(nom);note:='';
 repeat
   if n= length(listenoms) then note:='inconnu';
   if copy(LISTENOMS,n,1)= '[' then
           if nom=copy(LISTENOMS,n+1,L)
                then  note:=copy(LISTENOMS,n+1+L,2);
   n:=n+1
 until note<>'';
writeln(note)
end.
```

De la comparaison des deux programmes ci-dessus nous tirons trois conclusions:

- du point de vue de la simplicité des moyens mis en oeuvre, il n'y a pas dans cet exemple, de supériorité de la solution applicative: la solution impérative utilise seulement le jeu de fonctions réduit et une itération.
- la structure de liste ne suppose pas nécessairement un algorithme récursif, en effet la récursivité est terminale, et s'interpréte comme l'itération

REPETER <actions> L ← SP L JUSQU'A <conditions>

- la structure de liste suppose un algorithme dont l'interprétation constitue un **fonctionnement mental différent** de la solution impérative. Elle impose le raisonnement suivant: *supposons que le nom cherché soit le premier, alors on sait trouver le résultat, sinon, on transforme le problème pour s'en rapprocher.* Il s'agit donc, comme dans l'emploi d'une assertion invariante dans la construction d'une itération, d'un *raisonnement sous hypothèse* nécessairement plus proche de la récurrence que la simple action répétée de l'itération.
- au contraire, l'interprétation du parcours itératif de la chaîne de caractères peut utiliser des *images de dispositifs matériels*, par exemple un crayon parcourant une suite de caractères sur le papier, et s'arrêtant sur le caractère spécial. Le raisonnement s'appuie davantage sur la temporalité de l'exécution.

Un exemple de parcours de liste ne se ramenant pas à l'itération

Dans le problème ci-dessus, la structuration en liste ne montre pas sa puissance car le contenu n'est pas structuré sur plusieurs niveaux. Pour trouver des problèmes où cette structuration se justifie, il faudrait aborder des domaines tels que le traitement d'expressions algébriques, ou la représentation de connaissances: dans ces domaines, les algorithmes de parcours ne se réduisent pas à l'itération, car il faut

atteindre tous les niveaux de la liste. Donnons un exemple adapté de [BOURBION 84], ouvrage en principe destiné aux élèves de l'enseignement élémentaire et du début du secondaire (5-16 ans): *définir une procédure* SYMETRIE *qui, une procédure graphique étant donnée (par exemple* MAISON*), définit une nouvelle procédure graphique (*MAISONSYM*) de façon que l'exécution de* MAISONSYM *donne un graphisme symétrique de celui qui aurait été obtenu par l'exécution de* MAISON. Les définitions de procédure étant des listes, si l'on suppose que la procédure donnée n'appelle pas d'autres procédures, il suffit d'utiliser une fonction PERMUTE qui, une liste étant donnée, retourne cette liste donnée, toutes les occurrences de **TG** (qui désigne la primitive graphique: tourne à gauche) étant remplacés par **TD** (qui désigne la primitive graphique tourne à droite) et réciproquement. PERMUTE consiste en fait en un algorithme assez classique de transformation récursive d'une liste en profondeur:

```
si (PREM:L) est un MOT,on sait calculer la transformée,
sinon on s'y ramène, en utilisant la propriété de récurrence:
la transformée de la liste:L est la liste dont le PREM est
la transformée de PREM:L, et le SP la transformée de (SP:L).
```

Le *raisonnement sous hypothèse* porte ici sur une récurrence non linéaire: le processus par lequel on se rapproche du cas connu n'est pas une descente pas à pas comme dans l'exemple précédent, mais un processus par ramification. L'algorithme ne s'interpréte pas comme une itération. La structure de liste montre ici sa puissance, en même temps que l'interprétation du processus de calcul fait appel aux schémas mentaux de la récursivité dans toute leur généralité.

Conclusions

Nous pensons avoir montré que le choix des structures de données "textuelles" de **LOGO** (ou de **LISP**, **SCHEME** ou **PROLOG**) conduit à des raisonnements supposant un fonctionnement mental différent de celui qui est nécessaire pour un parcours itératif de chaîne de caractères. Dans le cas d'un parcours linéaire, on est en fait en présence d'un fonctionnement mental proche de celui qui est proposé par [ARSAC 80] (voir le sous-chapitre 1 de cette annexe) pour la construction de l'itération, dont nous avons dit qu'il est lui-même du niveau du raisonnement par récurrence. Il y a donc équivalence, **du point de vue des problèmes qui peuvent être traités**, entre le parcours itératif d'une chaîne de caractères, et le traitement en récursivité linéaire d'une liste "plate"; **du point de vue du fonctionnement mental**, le traitement de liste impose d'emblée un niveau de raisonnement équivalent au raisonnement par récurrence, alors que l'acquisition de ce niveau de raisonnement est, en initiation, un objectif, et non un pré-requis. Nous avons vu que le parcours itératif d'une chaîne se prête mieux à l'interprétation comme action répétée, et peut s'appuyer sur des représentations mentales plus proches d'un dispositif matériel.

Nous avons vu également que l'absence de typage des objets, qui est un autre choix de **LOGO,** peut poser des questions de cohérence du langage, et apparaît comme un artifice destiné à contourner une difficulté cognitive.

5. Note historique: l'utilisation des chaînes de caractères; conséquences pour l'enseignement

Il est intéressant, pour apprécier la place d'une notion dans un champ scientifique, de situer historiquement ses conditions d'apparition, et les applications qu'elle a permis. Concernant la notion de chaîne de caractères, ceci peut être fait en

examinant l'apparition de langages traitant cette structure, et les problèmes qui ont conduit à la réalisation de ces langages.

COMIT (1957) a été le «*premier langage de programmation significatif pour la manipulation de chaînes de caractères et la reconnaissance de combinaisons de lettres (...) Bien que destiné au départ à des applications de linguistique,* **COMIT** *a été utilisé pour des problèmes de manipulation formelle et de recherche d'information*». ([MORVAN 81][12]

SNOBOL (1962) est apparu avec «*l'ambition initiale était de proposer un langage meilleur que* **COMIT**.(...) *La puissance de ce langage* (**SNOBOL** 4) *a conduit de nombreux utilisateurs à l'employer pour de nombreuses applications de traitements de textes, au détriment du langage* **COMIT**» (ibidem).

Ces précisions (confirmées dans [SAMMET 69][13] et [GRISWOLD 75][14]) permettent de penser que les applications pratiques construites avec ces langages allaient de la traduction automatique, et de l'étude de texte à la manipulation symbolique, l'utilisateur pouvant ne pas être spécialiste de programmation. On sait que de telles utilisations ont marqué le pas: la traduction automatique, par exemple, s'est heurtée à des difficultés conceptuelles importantes et non-prévues au départ. La manipulation symbolique s'est orientée vers l'utilisation de systèmes où l'information est structurée plus en profondeur. Ainsi, [VIVET 84][15] présente un système de manipulation algébrique utilisant l'inférence; l'information y est organisée dans une «*base de connaissance*» constituée d'éléments hétérogènes («*résultats connus*», «*filtres et plans*»...) s'exprimant comme des «*régles de réecriture*» codées sous forme de `listes`.

Dans l'enseignement secondaire, le nombre important et la diversité des fonctions sur les chaînes de caractères en **LSE**. a été justifié par la volonté de développer des applications pour les domaines linguistiques et littéraires: «*..elles* (les fonctions **LSE**) *ont été conçues spécialement pour la manipulation aisée de textes. Ce qui dans l'enseignement des matières littéraires est primordial*» [CANAL 83]. Plus précisément, des tentatives ont été faites dans le domaine de l'analyse lexicographique.[16] Nous observons que de telles applications ne se sont pas répandues notablement chez les utilisateurs, et qu'elles se heurtent à l'absence totale de possibilité d'extraire des indications d'ordre sémantique d'un texte représenté en machine comme une suite de caractères.

L'ensemble de ces indications permet de comprendre pourquoi les chaînes de caractères ne constituent pas une partie importante des cursus d'informatique. L'utilisation des chaînes est, dans de nombreux cursus, réduite aux entrées-sorties, ou

---

12**Morvan & alter** «Dictionnaire de l'informatique» *Larousse* 1981

13**J.E. Sammet** «Programming languages ; History & Fondamentals» *Prentice-Hall* 1969

14**R.GRISWOLD** «String and list processing in SNOBOL4» *Prentice-Hall* 1975

15**M.Vivet** «CAMELIA A knowledge based mathematical system» in *ECAI.84 Advances in Artificial Intelligence (North-Holland)* 1984

16pour une présentation de ces méthodes, voir les articles de P. Muller, par exemple [MULLER 87]: «La désignation des personnages dans le tartuffe de Molière» *bulletin de l'association E.P.I.* n°47

à quelques exercices-types ("retournement" d'une chaîne, comptage d'occurrences...). Nous les considérerons pour notre part, comme une première approche de la structuration des données, provisoire, dans l'attente de structurations plus puissantes; c'est pourquoi, l'acquisition d'un jeu complet et sophistiqué de fonctions tel que celui qui est présent en **LSE** ne nous paraît pas essentiel; nous nous bornerons à l'acquisition du jeu réduit de fonctions que nous avons indiqué, et à la construction de la fonction `position`.

Annexes au chapitre 6

On trouvera dans ces annexes le texte de l'épreuve EPR1 (sous-chapitre 1.), un dépouillement question par question des réponses des élèves (sous-chapitre 2.), puis le texte commenté de trois séries d'entretiens avec certains élèves. Par entretien, nous désignons une séance de travail sous notre direction, en présence de l'ordinateur. La première série (paragraphes 3.1 et 3.2) concerne des élèves ayant de grandes difficultés, dont les plus évidentes concernent le langage *(D.Langage)* La seconde série (paragraphes 4.1 à 4.3) concerne des élèves maîtrisant la syntaxe, mais rencontrant des difficultés avec la structure de chaîne (*D.Sem.Cha.* et *D.Ord.Card.*) et le typage des objets (*D.Type*). Le dernier entretien (sous-chapitre 5.) concerne un élève qui réussit bien.

1. Le texte de l'épreuve

Rappel :
MID$(X$,a,b) où X$ est une variable chaîne, a et b deux nombres, est la chaîne constituée de b caractère pris dans X$, à partir du a-ième caractère.
Exemple :
Ce programme affiche "NJOU"

```
10 LET X$="BONJOUR"
20 LET Y$=MID$(X$,3,4)
30 PRINT Y$
40 END
```

question 1

On suppose qu'on a écrit le programme suivant :

```
10 INPUT X$ : REM entrée d'une chaîne au clavier
20 LET A$=MID$(X$,1,1)
30 LET L=LEN(X$)
40 LET B$=MID$(X$,L,1)
50 PRINT A$;B$
60 END
```

Indique ce que l'ordinateur affiche à la ligne 50,
1.a dans le cas où l'utilisateur a entré la chaîne "BONJOUR" à la ligne 10 .
1.b dans le cas où l'utilisateur a entré la chaîne "B" à la ligne 10 .

question 2

Ecris un programme pour que l'utilisateur, ayant entré une chaîne au clavier, l'ordinateur affiche la chaîne en passant la première lettre à la fin.
Exemple : si l'utilisateur entre "BONJOUR", l'ordinateur affichera "ONJOURB".

question 3

Complète le programme ci-dessous (ligne 30) pour que, l'utilisateur ayant entré un nombre entre 1 et 7, l'ordinateur affiche la note correspondante de la gamme (SO au lieu de SOL):

```
10 LET NOTE$="DO*RE*MI*FA*SO*LA*SI"
20 INPUT I
30: PRINT MID$(NOTE$,...,2)
```

question 4

Ecris un programme pour que l'utilisateur ayant entré une chaîne au clavier, l'ordinateur affiche le mot en intervertissant la première et la dernière lettre. Exemple : si l'utilisateur entre "SALUT", l'ordinateur affichera "TALUS".

question 5

Ecris un programme pour que, l'utilisateur ayant entré un nombre au clavier, l'ordinateur affiche le chiffre des unités.

2. Dépouillement de l'épreuve sur papier

De façon donner à une forme synthétique à ce dépouillement, et à faciliter la lecture, nous avons regroupé les réponses des élèves par questions. Pour chaque réponse, si elle est exacte nous indiquons à laquelle des réponses données dans la présentation de l'épreuve elle correspond, sinon nous donnons la description du type d'erreur, accompagnée si nécessaire de la réponse précise.

Question 1

8 élèves donnent les réponses attendues (`BR` puis `BB`).

2 élèves (**Florence P.** et **Alexandra J.**) donnent seulement `B` pour l'entrée "BONJOUR" ; **Alexandra J.** donne cependant `BB` pour l'entrée "`B`". Il peut s'agir d'une difficulté à interpréter l'ordinal `L`, comme argument de la fonction "sous-chaîne". 1 élève (**Bertrand G.**) donne "`B R`" pour l'entrée "BONJOUR", mais `B` pour l'entrée "`B`". Il peut s'agir ici plutôt d'une difficulté (*D.Sem.Cha*) à envisager le programme tel qu'il fonctionne réellement ; l'élève considérerait, à la suite de sa première réponse que le programme "retire la partie centrale", et appliquerait cette interprétation à la seconde entrée.

Questions 2 et 4 : Dépouillement élève par élève

Les questions 2 et 4 comportent les mêmes difficultés, c'est pourquoi nous les avons regroupées. Nous avions noté dans la présentation de l'épreuve que les élèves peuvent donner la réponse soit sous forme d'une seule expression (assez lourde, surtout pour la question 4), soit employer, pour chacune des sous-chaînes une affectation à une variable intermédiaire ; dans le second cas, il peuvent choisir soit de faire afficher chacune des parties de la chaîne dans l'ordre et sans passage à la ligne, ou de calculer le résultat par concaténation.

Stéphanie G.

question 2 : `LET A$=MID$(X$,2,L + MID$(1,1))`

Pas d'affectation de résultats intermédiaires ; le résultat se traduit par l'écriture d'une composée complexe de fonctions comportant des erreurs de syntaxe, en particulier, de parenthésage : (difficulté *D.Langage*)

question 4 : `LET A$=MID$(X$,2,L + MID$(1,1)`

```
LET A$=MID$(X$,1,L-1 + MID$(5,1))
```

On retrouve les erreurs de composition de fonctions, mais la deuxième ligne laisse penser à une confusion entre A$ et X$ (on fait une opération de transformation sur X$, puis une deuxième opération) donc à une difficulté *D.Sem.Cha.* Par ailleurs, la difficulté (*D.Ord.Card.*) à calculer le rang du dernier élément en fonction de la longueur se traduit par le choix de la constante 5 qui n'est valable que pour une chaîne de longueur 5.

Alexandra J. Florence P.

question 2 : LET A$=MIDS(X$,2,(LEN X$ -1) + (X$,1,1)

sans être identique à celle de **Stéphanie G.**, la réponse comporte des erreurs de syntaxe analogues. La question 4 n'est pas traitée.

Pierre E.

Décomposition à l'aide de l'affectation des résultats intermédiaires et l'instruction PRINT sans passage à la ligne. La question 2 est exacte. La question 4 comporte des erreurs dans le calcul de la longueur de la sous-chaîne extraite : L-1 pour la longueur de la partie centrale au lieu de L-2. L pour la longueur du dernier caractère.

Il y a manifestement dans la question 4 (mais pas dans la question 2) confusion (de type *D.Ord.Card.*) entre la longueur de la sous-chaîne extraite, et la position du dernier caractère de cette sous-chaîne.

Frédéric P.

Décomposition à l'aide de l'affectation des résultats intermédiaires ; affichage successif des sous-chaînes.

Les réponses sont correctes.

Pierre P.

Décomposition à l'aide de l'affectation des résultats intermédiaires ; affichage successif des sous-chaînes.

La question 2 est correcte ; la question 4 n'est pas traitée.

Bernard G.

Décomposition à l'aide de l'affectation des résultats intermédiaires affichage successif des sous-chaînes.

La question 2 est correcte syntaxiquement ; le mot est décomposé en deux parties, A$ et B$, concaténées à la fin dans cet ordre, mais si A$ est bien la sous-chaîne à partir de la seconde lettre, B$ est en fait la dernière lettre du mot ; il y a donc une erreur liée à une difficulté *D.Sem.Cha.*, le deuxième argument étant compris comme la position du caractère dans le résultat, et non la position au moment du calcul.

La question 4 n'est pas comprise : l'élève se donne pour tâche de retourner la chaîne au lieu d'inverser seulement le premier et le dernier caractère ; il donne une solution comportant un nombre variable de lignes, affectant à des variables A$, B$,...., successivement chaque caractère ; le résultat est donné par une instruction d'affichage de longueur variable ; les points de suspension sont employés un peu comme dans certains raisonnements mathématiques, mais le programme n'est pas exécutable.

Armelle D. Karine C. **(réponses identiques)**

Décomposition à l'aide de l'affectation des résultats intermédiaires sauf pour `LEN(X$)` ; affichage successif des sous-chaînes.

La question 2 est correcte ; la question 4 comporte une erreur de calcul d'une longueur (longueur de la sous-chaîne centrale : `LEN(X$)-1`) et d'une position (position du dernier caractère : `LEN(X$)-1`). (*D. Ord.Card.*)

Caroline C.

Décomposition à l'aide de l'affectation des résultats intermédiaires ; affichage successif des sous-chaînes. La question 2 est correcte, à l'exception de l'absence d'une instruction d'affichage `PRINT` à la ligne 40 (*D.Langage*) . La réponse à la question 4

```
LET A$ = MID$(X$,2,LEN-1)
LET B$ = MID$(X$,LEN,LEN-1)
PRINT A$;B$
```

se présente comme l'extraction syntaxiquement correcte de deux sous-chaînes `A$`, puis `B$` ; `A$` pourrait être la sous-chaîne centrale (mais alors, il y aurait une erreur sur la longueur), et `B$` le dernier caractère (erreur également sur la longueur), mais il y a plusieurs erreurs liées à (*D.Sem.Cha.*): les arguments de la fonction `MID$` paraissent mal compris, les éléments de la chaîne `A$` et `B$` sont restitués dans l'ordre initial, et le premier caractère n'est pas pris en compte.

Florent R.:

question 2 : la seconde partie du mot est calculée correctement (ligne 30: `LET A$=MID$(X$,2,L)`). Par contre la ligne 20 `SCH("X$",1,1)` laisse penser à une traduction directe d'une formulation du type "on prend une lettre dans `X$`, à partir de la première", sans que nécessairement cette action soit comprise comme modifiant `X$`. (*D.Langage*)

Les lignes 40 `LGR(SCH("X$,1,1)+("A$)` et 50 `PRINT A$;X$` laissent penser que, pour cet élève, après la ligne 40, `X$` contient la première lettre, la ligne 40 s'interprétant difficilement : on pense à une phrase du type : la longueur (c'est à dire le mot tout entier), est le résultat de l'extraction du premier caractère concaténé avec `A$` , ce qui aurait plutôt valeur de commentaire (*D.Sem.Cha.*).

La question 4 n'est pas traitée.

Question 3

Les élèves ont à compléter la ligne 30: `PRINT MID$(NOTE$,...,2)` , les "..." étant à remplacer par une expression fonction de `I`.

5 élèves (**Armelle D., Karine C., Pierre P., Frédéric P., Bernard G.**) produisent la fonction attendue.

Sur deux autres copies (**Pierre.E., Stéphanie G.**), on trouve des traces de comptage (1, 4, 7, 10,...) attestant de la recherche infructueuse d'une loi.

En plus de la ligne 30 attendue, **Bernard G.** propose un programme qui ignore totalement la chaîne `NOTE$`, et se compose de 7 lignes d'instructions conditionnelles du type : `IF I=1 THEN PLAY "DO"` ...

Question 5

Les trois élèves qui répondent utilisent les fonctions "longueur" et "sous-chaîne" ; l'utilisation est correcte pour l'un (**Bernard G.**), et comporte dans les 2 autres cas (**Armelle D., Karine C.**) l'erreur (liée à *D.Ord.Card.*) sur la position du dernier caractère de la chaîne, déjà constatée à l'exercice 4.

Par contre, ces 3 élèves font porter les fonctions sur des variables numériques (identificateurs non terminés par un "$", en **BASIC**), ce qui constitue une erreur de type.

3. La conception naïve du langage de programmation (entretien avec Stéphanie B. et Stéphanie G.)

Les élèves sont devant l'ordinateur, avec le résultat de leur recherche écrite ; nous enregistrons au magnétophone, prenons des notes et relançons la recherche. Le dialogue enregistré est à droite, les commentaires à gauche. Nos interventions sont en italiques . Les remarques des élèves,sont précédées de leurs initiales (**S.B.** et **S.G**).

L'entretien s'est déroulé en deux phases ; l'une d'une demi-heure porte sur l'exercice 2 ; l'autre, d'une heure la séance suivante, porte sur l'exercice 4. Nous avons vu en effet dans le dépouillement de l'épreuve sur papier que la réponse de Stéphanie G. à l'exercice 2 comprend de nombreuses erreurs de syntaxe, manifestant une incompréhension des contraintes du langage ; il est donc intéressant de chercher à en savoir plus sur ces difficultés, et leur éventuelle persistance dans l'exercice 4.

Stéphanie B. (**S.B.** dans la suite) était absente lors de la passation de l'épreuve écrite. Elle forme un groupe avec **Stéphanie G.** (**S.G.** dans la suite) lors des travaux sur ordinateur, et rencontre les mêmes difficultés.

3.1 Première phase : résolution de l'exercice 2

S.G. explique à **S.B.** sa réponse à la question 1.

S.G. Là t'as INPUT, là "Bonjour", là tu prends une lettre à partir de la première, après t'as la longueur totale ça c'est la longueur totale, et 1, la première lettre , donc PRINT ça va être A$ et B$ ça va donner "B".

• *"B"?*

S.G. Ah! oui, "BR", la longueur totale, et on prend 1.

• *ça donne la dernière lettre*

S.G. ah oui, la dernière lettre

• donc "BR" ; alors pourquoi est-ce qu'ici ça me donne "BB"? là où l'utilisateur a entré "B"?

S.G. ben parce-qu'il ne peut pas être égal à autre chose que "B" !

• *oui c'est à la fois la première et ...*

S.G. S.B. et la dernière.

réponse de S.G. à l'épreuve écrite :

```
10 INPUT X$ : REM "BONJOUR"
20 LET
A$=MID$(x$,2,L+MID$(1,1))
30 PRINT A$
40 END
```

• *(lit l'énoncé de la question 2)*

.....

• *il y a des petits problèmes de syntaxe; tu les vois...*

S.G. ah oui, il manque des parenthèses..

L n'a pas reçu de valeur dans le programme. On pourrait insérer une ligne telle que

```
15 LET L = LEN(X$)
```

Mais la correction que propose **S.G.** consiste à remplacer L dans l'expression de la ligne 30 par LEN(X$), ce qui la complique encore.

• *qu'est-ce que c'est "L"?*

S.G. L c'est la longueur, ... faudrait que je lui marque LEN ?

• *LEN de quoi ?*

S.G. LEN ...

S.B. ben LEN de 2?

S.G. de X$?

• *oui, il ne peut pas le deviner !*

... • *il y a des choses qui ne vont pas ; MID$, ça a combien d'arguments? je le dis parce que on va avoir une erreur de syntaxe; MID$, il prend combien d'arguments ? ... c'est à dire le nombre d'objets qu'il y a derrière la parenthèse.*

S.G. ben, un seul

Le mot "argument" n'est pas connu des élèves.

• *moi, j'en vois 3, je vois* X$, *je vois a, je vois b ;* X$ *est une chaîne, a et b sont des nombres..*

S.G. ah oui...

• *tu en as combien, là...il faudrait s'arrêter derrière* LEN(X$).

S.G. il faudrait mettre X$...

• *il faudrait mettre une parenthèse derrière... il faut terminer* MID$...

S.B. toutes ces parenthèses, ça me rappelle les maths...

• *bon alors,* MID$, *qu'est-ce-qu'il fait là maintenant ?(* celui de la fin de la ligne)

S.G. faut le mettre dans une autre ligne ?

• *oui, mais tu avais mis un "+", ici, qu'est-ce qu'il signifie ?*

S.G. ben, là .. comment.. j'avais écrit "BONJOUR", donc X$, à partir de la deuxième, on prend la longueur totale, plus la première lettre, on en prend une.

• *c'est pour cela que tu l'avais mise dans la parenthèse ?*

S.G. ben oui...

S.B. je ne suis pas au courant du tout..

• *tu vois pas du tout? comment tu l'aurais fait?*

S.B. comme ça, euh.. ben faut que je revoie le programme.

• MID$, *tu l'aurais mis dans la parenthèse, comme ça? parce que tu voulais reprendre..*

S.B. c'est faux, de toute façon ces parenthèses là

S.G. ben oui...

• *c'est faux cette parenthèse là ? c'est à dire que vous auriez bien vu quelque chose qui soit comme cela* (rétablit sur le papier les parenthèses correctement balancées)

S.G.S.B. (rétablissement de parenthèses)

• *c'est pas la peine qu'on essaye, il va y avoir une erreur de syntaxe ; ce que je voudrais savoir c'est un peu... comment on comprend cela ... là tu aurais mis un "+", c'est ça*

S.B. ah non, je n'aurais pas mis un "+", moi

• *qu'est ce que ça peut vouloir dire* LEN(X$) *et tout de suite après,* MID$(1,1)*? c'est quoi,* MID$(1,1)*? qu'est ce que ça représente?*

S.B. c'est pour...

S.G. ben, ça représente "B"

• *la première lettre..on aurait pu dire "SALUT", ça représenterait sa première lettre. il y a quand même un petit problème, c'est que* MID$ *n'a que deux arguments ... Parce que, il ne se rappelle pas forcément que c'est* X$..

S.G. LEN de X$ et 1 et 1

• *oui mais, c'est de* MID$ *dont je parle, il n'a que deux arguments , il devrait en avoir 3*

S.G. eh ben, là on fait X$,1,1

• *oui, parce qu'il ne se rappelle pas forcément ...*

S.G. là c'est bon comme ça?

• *il y a un petit problème, c'est que là tu n'as pas de parenthèse fermante...*

S.G. oui...

• *là tu es tombée sur des problèmes de parenthésages qui ne sont pas très drôles*

S.B. ça n'est pas drôle en général !

La ligne 20 telle que **S.G.** l'a écrite est en fait cohérente, de son point de vue : l'écriture L+MID$(1,1)constitue un troisième argument du premier MID$ cohérent avec la façon dont cette élève considère le langage. L (longueur) est compris comme un ordinal (on prend toute la longueur, donc jusqu'au dernier) et non comme un cardinal. Mais en même temps L est considéré comme le codage d'une action ("on rend la longueur totale") et non comme un véritable nombre. Le fait de donner aux fonctions le statut d'actions permet à 'élève d'additionner un nombre et une chaîne, faisant ainsi une synthèse des opérations de concaténation des chaînes et d'addition des nombres toutes les deux désignées par le signe + : "on prend la longueur totale (nombre) plus la première lettre (chaîne)".

La compréhension des fonctions comme codage d'actions conduit **S.G.** à ne pas répéter l'argument. Cet argument (ici X$) est en effet compris comme l'objet sur lequel on agit, et que par conséquent, il n'est pas nécessaire de repréciser. Nous interprétons cette non-répétition comme l'utilisation d'arguments implicites, ce qui est une interprétation sans doute trop syntaxique.

Encore une recherche des difficultés de l'élève dans la syntaxe : l'expression de la ligne 20 de la réponse de **S.G.** est en fait correctement parenthésée.

• *il faut autant de parenthèses qui ferment, que de parenthèses ouvrantes*

S.B. j'ai compris ça

• *là il y a encore un problème le MID$,*

S.B. il manque le 3!

• *il manque le 3? celui là je ne lui vois pas 3 arguments, parce que …*

S.G. ah oui, il faut une virgule là

• *il faudrait une parenthèse, pour terminer si tu veux, le troisième argument.. j'ai* MID$, *un argument qui est une chaîne de caractère, un argument qui est un nombre, et un argument qui est un nombre ; c'est* LEN(X$), *comme ça? tu vois, cette parenthèse ferme un bloc ; celle là correspond à celle là,... j'ai bien un bon bloc :* MID$(X$,1,1) *. Il n'y a plus d'implicite comme tu faisais, mais en même temps, on a séparé ces deux* MID$, *mais il faudrait peut-être les rejoindre*

S.B. c'est "+"

• *c'est "+", que t'attendais ; mais en fait, à quoi il sert le "+"?*

S.G. eh ben, pour que la chaîne de caractère soit unique

• *c'est ça, pour faire une seule chaîne de caractère*

S.B. oui, ça marche.. c'est cela

... (exécution)

• *essaie de reprendre le 4, en te servant de ce que tu as trouvé pour le 2*

3.2 Seconde phase (la semaine suivante) : résolution de l'exercice 4.

rappels sur la question 2.

le programme écrit par **S.G.** lors de l'épreuve écrite :

```
10 INPUT X$ : REM "SALUT"
20LET A$=MID$(X$,2,L+MID$(1,1)
30 LET A$=MID$(X$ 1,L-
1+MID$(5,1))
```

Au début du travail sur ordinateur, les élèves corrigent spontanément le programme écrit par **S.G.** à l'épreuve écrite, en s'inspirant des corrections apportées pour le problème **ONJOURB** à la phase précédente. Le programme suivant est entré sur l'ordinateur:

```
10 INPUT X : REM BONJOUR
20 LET A$=MID$(X$,2,LEN(X$))+
MID$(X$,1,1)
30 LET B$=MID$(X$,1,LEN(X$)-
1)+ MID$(X$,1,LEN(X$)-1)
```

La ligne 20 prend en compte les corrections apportées lors du premier entretien à la solution de l'exercice 2. Les élèves ont apporté également des corrections à la ligne 30, qui se révèle correcte syntaxiquement.

Concernant cette ligne 30, la seule évolution notable, pouvant être interprété comme une évolution de la compréhension, est l'introduction d'une seconde variable B$ à laquelle est affecté le résultat de la ligne 30, alors que sur la copie de **S.G.**, les deux résultats étaient affectés successivement à A$.

• *le programme qu'on avait fait consiste à faire passer la première lettre à la fin. Bon , on avait repris un peu le programme qu'elle avait fait, mais on avait changé un peu...*

S.G. oui, les parenthèses...

• *est-ce que vous voulez faire la question 4 ; en principe vous devriez savoir le faire... Il s'agit d'inverser la première et la dernière lettre.*

S.G. Dites nous plutôt comment on fait...

S.G. moi j'avais fait ça (référence à la copie)

S.B. bon alors on y va ...

• *bon , on le fait tourner... ... tu donnes un mot...*

Exécution...Pas de résultat

...

• *on n'a rien fait afficher!*

Il n'y a pas d'affichage ; en fait les affectations à A$ et B$ sont là pour la correction syntaxique, mais A$ et B$ n'ont pas valeur de résultat; les élèves ne leur accordent pas de signification et ne les affichent pas.

S.B. qu'est-ce qu'on aurait pu afficher ?

• A$...

S.G. ah oui INPUT !

• *ah non* PRINT *plutôt ; à la ligne 40, par exemple*

S.G. oui, et à la ligne 30, il faut faire l'inverse

Il n'est pas sûr que pour les élèves, le résultat soit A$+B$, mais suite à nos suggestions, elles ajoutent une ligne 40 :

```
40 PRINT A$+B$
```

• *c'est cela, passer la dernière lettre avant la première*

• *qu'est-ce qu'on fait à la ligne 20 ?*

S.G. on prend la première lettre pour la mettre à la fin...

Exécution ; Résultat :

```
ONJOURBBONJOUBONJOU
```

S.G. j'ai fait un programme du tonnerre...

Nous pensons sur le moment que l'amélioration de la syntaxe constitue un progrès vers la solution, en ce sens que les élèves vont pouvoir interpréter les résultats à l'exécution

• *c'est correct du point de vue de la syntaxe ; vous avez fait des progrès!*

• *si on faisait afficher* B$ *tout seul, qu'est-ce qu'on obtiendrait?*

S.G. ça me ferait passer, euh..

• *dans le résultat, qu'est ce qui appartient à* A$ *et qu'est-ce qui appartient à* B$*?*

S.G. ce qui appartient à A$, c'est cela (ONJOURB), et ça à B$..

En fait pour les élèves, chacune des lignes 20 et 30 code une action: pour la ligne 20, c'est la même action qu'à l'exercice 2, c'est à dire "passer la première lettre à la fin", et la ligne 20 de l'exercice 2 est recopiée sans modification ; pour la ligne 30, c'est "passer la dernière lettre au début", action que l'entretien ne permet pas d'analyser.

• *il y a deux fois la même chose dans* B$...

Référence à des exercices sur les ensembles ???

S.G. ah oui..donc INTER!

S.B. ah non ça ne marchera pas ...

• *vous voulez faire une intersection ?*

S.G. ben oui...

• *ce qu'il faudrait faire c'est des parties... quelles sont les parties qui nous intéressent?*

S.G. ben , la première lettre...la longueur totale - 1 et puis la longueur totale - 1,

• *la première lettre, d'accord..*

S.B. ben oui, parce que bon, d'accord, mais là ça va pas...

Il n'est pas clair du tout pour les élèves que les fonctions produisent des résultats et que la fonction de concaténation permet de "recomposer" ces résultats pour obtenir la solution souhaitée.

• *il s'agit d'intervertir les deux lettres, donc ce qu'on doit avoir, c'est la première lettre, puis cette partie là, comment on pourrait l'appeler ?*

S.G. c'est le R

La décomposition en trois parties est imposée par nous sans réelle adhésion des élèves.

• *d'accord, c'est un R, mais c'est la dernière lettre, et cette partie là c'est le corps ...ce qu'il faudrait, c'est les repérer*

• *bon on va mettre* A$= *par exemple, la première lettre ; est-ce qu'on sait le faire cela, trouver la première lettre?...*

S.B. euh... oui..

• *mettre* B$ = ...

S.G. X$...

• *le corps... ça, on ne sait pas encore le faire... et mettre* C$ =... *la fin. Supposons qu'on sache faire cela...*

S.G. il faudra faire un affichage

• *oui qu'est-ce qu'on mettra en ligne 40?*

S.G. ben A$-C$...

• *ça n'existe pas ! + veut dire : on les met l'un à côté de l'autre.*

S.B. ben + !

• *bon, si je fais* A$+B$, *c'est cela?*

S.B. non A$+C$!

S.G. ça va donner la première lettre et la dernière !

S.B. bon alors A$ + C$ +...

S.G. ou alors A$ = B$ = C$ ...

• *égale* C$ *??? ils sont différents, ils ne peuvent pas être égal...*

Proposition de restitution dans l'ordre initial.

S.B. c'est A$ + B$ + C$!!

• *d'accord , on essaie qu'est-ce que ça va donner ?*

S.G. ça va donner la même chose que ça, oui

S.B. bon, eh bien, C$ + A$ + ...

Propositions diverses difficiles à interpréter.

S.G. non C$ facteur de A$ + B$

• C$ + ...

S.G. C$ facteur de A$ + B$

• *il y a +, il n'y a pas facteur... on ne multiplie pas les chaînes de caractère, on les ..., on les concatène.*

S.G. faut mettre entre parenthèses A$ +....

S.G. ah oui, faut mettre B$ + C$ + A$

• *vous allez les avoir tous donné là!*

L'ordre des opérandes dans la concaténation n'est pas perçu comme critique par beaucoup d'élèves; Nous verrons d'autres exemples au chapitre 8.

S.G. on est nul alors !

S.B. moi je comprends rien !

• *qu'est-ce qu'il veut dire le + là ?*

S.B. le +, euh... euh... mettre côte à côte...

• *mettre côte à côte, oui. Si je fais, euh..., par exemple, euh..Alain + Michel* (écrit sur papier sans guillemets) ,

S.B. ça donne AlainMichel (pas possible de savoir la typographie éventuelle)

Si je fais Michel + Alain ?

S.B. ça fait MichelAlain

• *d'accord, c'est pas indifférent l'ordre*

S.B. ben non...

• *si je fais* A$ + B$ + *C$, ça fait...*

S.B. oui...

• *B..ONJOU..R.... si je fais* C$ + A$ + B$, *ça fait...*

S.B. JOURBON, ou je sais pas quoi...

vague idée d'un retournement

• *oui ça va faire R..BONJOU ... changer la première et la dernière, c'est pas plus compliqué! si j'ai* A$ + B$ + C$ *qui est le mot dans le bon ordre,*

S.B. oui...

S.G. ben, je sais pas moi... en fait le corps, c'est B$... donc on va mettre C$ avant B$

• *oui*

S.G. et A$ après

• *oui*

La simple écriture C$+B$+A$ semble laisser **S.G.** insatisfaite, car cette écriture ne comporte pas d'actions telles que "passer une lettre..."

S.B. c'est ça ?

• *on a supposé qu'on savait là trouver A$, B$ et C$; maintenant, on va dire comment on les trouve...*

S.G. ben , en partant de la deuxième lettre, et en faisant la longueur totale moins 1..

• *tu saurais l'écrire là avec des MID$...*

S.G. oui, oui ça fait: 2,LEN(X$)-1

• *regarde là, si je prends BONJOUR, la longueur totale - 1 ça fait 5, alors, la cinquième lettre ...*

S.G. ben, c'est le R

La difficulté concernant la compréhension de la fonction "longueur" comme argument de MID$, que nous retrouverons et analyserons lors de l'entretien avec **Armelle D.** et **Karine C.**

• *un, deux, trois, quatre, cinq .*

S.B. plus 1

• *t'as dit la longueur totale moins 1 plus 1 ! y a quelque chose de plus simple*

... corrections au programme
Exécution. Résultat: RONJOURB

• ...LEN(X$)...

S.G. voilà ... il y a une lettre en trop?

• *il y a une lettre en trop , oui ! le R est toujours là! où est-ce qu'on s'est trompé?*

S.G. ben là! (ligne de calcul de C$?)

• *non le R est bien devant...*

S.G. oui.. ah la, la la la , on n'est pas des ordinateurs...

S.B. franchement des trucs comme cela, c'est pas le pied !

• *vous n'êtes pas des ordinateurs, c'est vous qui la dominez, au contraire la machine...*

S.G. la première..., là c'est bon cette ligne là...

• *j'ai C$ et le R est bien devant...*

S.G. non c'est la dernière,.. l'avant dernière...

• *oui..*

S.G. la longueur totale - 1, ben normalement, ça doit faire ça

• *prends un exemple précis*

S.G. 6 lettres pour BONJOUR.

• *moi, j'en compte 7 bon alors, la longueur totale - 1*

S.G. ben, y en reste 6

• *bon, je pars de la deuxième lettre, et j'en compte 6...*

S.G. faut partir de la première...

• *ce qu'on veut, c'est le milieu*

S.G. c'est -2 ...

• oui

Serait-il immoral de faire chercher les élèves en classe d'informatique ?

S.G. pourquoi vous ne nous l'avez pas dit ? hein...

• *ben c'est à vous de trouver...*

S.G. ça c'est pas mal, on me laisse marquer des erreurs...

... correction du programme...

S.G. voilà ben il est gentil

...(succès)

S.B. non, c'est fini, on veut plus...

4. Les difficultés avec les ordinaux; le choix du type(Entretien avec Armelle D. et Karine C)

Les réponses de ces deux élèves à l'épreuve, sont dans une bonne moyenne, comme le montre le tableau paragraphe 4.4 du chapitre 7, ce qui confirme leurs résultats habituels. Ces entretiens ont pour but d'approfondir principalement l'analyse des erreurs concernant la logique interne de la structure de chaîne *(D.Sem.Cha)* , et l'erreur de type *(D.Type)*.

L'entretien ci-dessous a lieu une semaine après la passation écrite de l'épreuve EPR1. Les élèves (Armelle D. et **Karine** C.) sont devant l'ordinateur, avec le résultat de leur recherche écrite ; l'auteur de cette thèse enregistre au magnétophone prend des notes et relance la recherche. L'entretien s'est déroulé sur une heure, en trois phases. La première phase est consacrée au passage sur machine du programme donné

en réponse à l'exercice 4 lors de l'épreuve, et porte surtout sur les difficultés ayant trait à la logique de la structure (*D.Sem.Cha).* La seconde phase est consacrée à un exercice du même type que l'exercice 3, et porte surtout sur les difficultés ayant trait aux éléments numériques (*D.Ord.Card.*). La troisième phase se rapporte au passage sur machine du programme donné en réponse à l'exercice 4, et conduit à étudier les difficultés ayant trait au typage des objets.

Le dialogue enregistré est à droite, les commentaires à gauche. Nos interventions sont en italiques, celles des élèves sont précédées de leurs initiales (respectivement par **A.D.** et **K.C.**)

4.1 Première phase : recherche de l'exercice 4 (RONJOUB)

Programme écrit par Armelle lors de la passation de l'épreuve:

```
10 INPUT X$
20 LET A$=MID$(X$,LEN(X$)-1,1)
30 LET B$=MID$(X$,2,LEN(X$)-1)
40 LET C$=MID$(X$,1,1)
50 PRINT A$;B$;C$
60 END
```

• *on fait le 4 ; il s'agissait d'inverser la 1ère et la dernière lettre*

A.D. On peut mettre "Bonjour"

K.C. Non c'était "SALUT"

(exécution... résultat UALUTS)

A.D. **K.C.** C'est pas ça

A.D. on cherche... C'est la que ça va pas ; on a pris le "U"; c'est le "T qu'il faut prendre

on note en ligne 20 l'erreur caractéristique sur la position du dernier caractères, et en ligne 30 une erreur sur la longueur de la sous-chaîne.

K.C. attend là quand on lui dit... il nous prend le "T"; c'est dans cet ordre là.. Il prend la longueur moins 1 ; à mon avis efface le -1; (dans la ligne 20) ben qu'est ce qu'il va faire de la longueur du mot? Je veux voir ce que ça donne. Ben déjà c'est meilleur

la ligne 20 est corrigée en :

```
20 LET A$=MID$(X$,LEN(X$),1)
```

(exécution... résultat TALUTS)

K.C. Déjà il nous retire le "U" ; dans l'autre truc, on lui demandais de prendre le "U" et de nous mettre le S à la place. Longueur y prend tout...

K.C. c'est inversé

A.D. oui t'as raison

A.D. y a juste le "T" qui va pas

K.C. retire -1 là (ligne 30)

la ligne 30 est corrigée en :

```
30 LET B$=MID$(X$,2,LEN(X$))
```

A.D. oui on va toujours voir ce que ça donne...

K.C. à mon avis c'est ça ...

pas de changement, car on prend un caractère de plus,... qui n'existe pas.

résultat : TALUTS

A.D. c'est toujours pas cela

K.C. n'empêche que ça lui a rien changé

retour à la version antérieure

A.D. faut peut être remettre -1

nous tentons d'obtenir une explication concernant l'écriture des fonctions, puis de faire exécuter le programme "à la main" (recherche des valeurs des fonctions pour une entrée donnée).

• *C'est clair ce 3ème paramètre là? il indique quoi? quand on a X$, a,b , alors a c'est le numéro du caractère à partir duquel on commence, et b c'est?*

A.D. c'est le nombre.., c'est combien on veut de lettres.

• *c'est le nombre de caractères... Ici* LEN(X$) *ça vaut combien?*

A.D. cinq

K.C. cinq oui.. on commence au 2ème, on en prend cinq ...

L'idée de **K.C.** est-elle que la ligne 20 a eu un effet de bord sur la chaîne X$?

K.C. ben là on l'a extrait le "T"; pourquoi il nous le met là ?

A.D. ben tu l'a pas retiré euh...c'est pas parce que tu l'as extrait là que...

• *c'est pas parce que tu l'a enlevé qu'il n'est plus dans* X$

A.D. K.C. donc il faudrait l'enlever...

• *dans quelle chaîne?*

A.D. c'est dans la deuxième à mon avis...

K.C. il faut avoir que "ALU"

nous invitons à revenir sur le calcul des valeurs numériques des fonctions impliquées par la recherche des sous-chaînes.

• *comment peut-on avoir "ALU"... à partir de* X$ *? qui n'a pas changé! c'est forcément les 3 lettres...*

A.D. en partant de "TALUS", comme si on prenait B$

• *on est d'accord, tu le prends dans "TALUS",* X$ *il n'a pas changé..*

...

• *on part bien du bon point, on ne sait pas combien il faut en prendre en fait, le "T" il est toujours là, pas de problème?*

A.D. ben, faut bien l'enlever!

K.C. mais comment?

• MID$ *ça veut dire plutôt "ne pas le prendre"*

A.D. ah...

K.C. tout à l'heure, qu'est-ce qu'on a écrit? ...

nous invitons à isoler le problème de la recherche de la sous-chaîne "corps", pour éviter les interférences avec un effet de bord supposé se produire lors de la recherche de la première sous-chaîne.

• *en fait, ce que vous pourriez faire, c'est un programme où on ait juste le corps sans le premier et le dernier ; juste afficher le mot sans le premier et le dernier donc, si je met "SALUT", ça doit afficher "ALU"*

A.D. on met B$...

• *oui, ou essayez d'écrire sur le papier, ça ira peut-être mieux... ... on commencerait avec* INPUT X$, *il faudrait faire une ligne dans laquelle on prenne seulement... oui 25 par exemple...*

A.D. a un doute concernant un éventuel effet de bord de la ligne 30 sur la suite du programme.

A.D. ça marche plus après... si on lui demande d'écrire B$, si on lui demande d'enlever les deux là, après là ça va plus marcher tout ça...

K.C. Si

A.D. qu'est ce qu'il va en faire du "S" et du "T"...

K.C. Ben il va l'enlever, quand on lui demande...

A.D. où il va aller le mettre?

• *Où il va les mettre ???*

A.D. on tourne en rond

• *il va les mettre ; on ne se pose pas la question ; en fait* X$, *il continue à exister … il n'est pas changé* X$.

La recherche de la sous-chaîne est comprise comme une opération de séparation d'éléments de X$, et, si la sous-chaîne corps est bien comprise comme affectée à B$, il y a doute sur ce que deviennent les autres produits de la séparation. **A.D.** conçoit, à ce moment les chaînes comme des objets physiques non duplicables.

A.D. ah d'accord

• *si on met en 55* "PRINT X$", *qu'est-ce-que vous allez avoir?*

A.D. il va réécrire "SALUT"

Nous invitons plus directement à isoler le problème de la recherche du "corps" du mot.

• *ben oui, en fait le "S" et le "T", il ne les sépare pas, tu vois? … en 29 je vais mettre* "END" *; entre 20 et 29, vous allez faire un programme qui va afficher le milieu*

il y a une confusion concernant le troisième argument de X$, qui est ici compris comme une position .

A.D. on lui met B$...donc il prend à partir de la 2ème lettre, après la longueur moins 1

K.C. tout à l'heure on l'avait?

A.D. faudrait peut-être mettre "PRINT"

• *faudrait peut-être faire imprimer quelque chose*

A.D. oui !

A.D. je mettrais "PRINT B$"...

A.D. ah non -1 ça marche pas! …

K.C. -2

Nous nous référons à un exercice de programmation antérieur à l'épreuve: isoler le radical d'un verbe afin de le conjuguer ; dans cet exercice, la position de départ de la sous-chaîne étant 1, la confusion position-longueur sur le troisième argument, ne porte pas à conséquence.

nouvel essai pour faire envisager un exemple numérique, cette fois sur un "grand nombre".

• *la différence, c'est que dans le cas du radical, vous partez de la première, tandis que là vous partez de la deuxième... Il ne s'agit pas d'une position, mais d'un nombre ; vous voulez vous arrêter à l'avant-dernière lettre Supposez que j'ai une chaîne de 1000 caractère, je veux retirer le premier et le dernier ; combien va t'il m'en rester?*

A.D. 998...

• *par rapport à* LEN(X$) *?*

A.D. K.C. -1, ... -2!

• *c'est normal, si tu démarres à partir du deuxième*

A.D. on va essayer

modification de la ligne 30 en

```
30 LET B$=MID$(MID$,2,
          LEN(X$)-2)
```

K.C. si ça marche, va falloir retirer le END.

A.D. on peut effacer aussi la 25

succès

K.C. c'est bon

A.D. on met "TALUS" ça va faire "SALUT".

• *ECOLE ça va faire quoi?*

A.D. oh ben ECOLE ; il réfléchit plus vite que moi, ça m'énerve

• *LEON*

A.D. LEON NOEL ; plus rapide que lui.

K.C. ah Non NEOL

• *on saura plus tard, à partir de là tout retourner...*

A.D. doucement...

• *est-ce que vous avez vu...*

A.D. on mettait -1, c'était -2, en fin de compte...

et pourquoi vous faisiez cette erreur là?

K.C. parce que on prenait, enfin moi ..

A.D. nous...

K.C. SALUT, on retirait la longueur du mot moins une lettre, donc on repartait en arrière pour prendre que le T,

A.D. alors qu'il y en avait une qui était venue en arrière déjà , je crois que c'est celle là.

• *vous pensiez que ayant pris la dernière lettre, vous ne l'auriez plus...*

K.C. oui c'est comme ça

Il n'est pas facile d'obtenir une explication de la part des élèves, surtout a-posteriori ; il semble que LEN(X$)-1 code en fait une action: "prendre" (ou "écarter" le dernier caractère).

• *ou est-ce qu'il y a une confusion par rapport à la position, c'est à dire que vous pensiez je vais de la deuxième à l'avant dernière.*

A.D. Non non , je crois que ... -1, on prend le "T", c'est la longueur moins la dernière lettre...

• *toi tu pensais plutôt que c'est parce qu'il est déjà pris le "T", donc que de toute façon on ne l'aurait pas...*

Il semble bien que cette élève reste fixée sur son interprétation initiale, et attribue la nécessité de l'argument LEN(X$)-2 (au lieu de sa proposition initiale LEN(X$)-1), à la présence de la première lettre à la fin du mot, qui serait un effet de bord du premier appel de MID$.

K.C. non moi je pensais comme j'avais fait, je pensais pas d'abord qu'on avait pris le "S" qu'on l'avait ajouté...

4.2 Seconde phase : résolution d'un exercice prolongeant l'exercice 3 (NOTE)

Il s'agit de contrôler la compréhension de la solution (correcte) trouvée par les élèves lors de la passation écrite.

• *je peux vous demander autre chose...de faire quelque chose un peu du même genre* (que l'exercice 3) *au lieu de cela, de faire l'alphabet, c'est à dire qu'il faudrait que si je tape "3", par exemple, j'ai "C", que si je tape 26, j'ai "Z"...*

La chaîne constante, ainsi que la longueur du résultat (1 au lieu de 2) sont trouvés spontanément. La formule trouvée à l'exercice 3 et concernant la position de départ de la sous-chaîne extraite est essayée mentalement, et non sur machine, et invalidée sur une valeur particulière (5).

A.D. c'est cela, c'est le même type que celui-là ; il va falloir écrire l'alphabet, là...

• *pas de problème...*

A.D. on y va...

K.C. pourquoi alphabet... tu veux dire alpha$?

L'expression :"faut trouver un rapport..", ainsi que l'exclamation "Pourquoi 3? deux deux.." laisse penser que le rapport d'homothétie est compris comme inhérent au problème, et ne résulte pas d'un essai au hasard, validé par une expérience.

A.D. alors, faut pas donner la même valeur que la dernière fois (à la ligne 30) virgule, na.na. na....,virgule 1; maintenant il manque le milieu.. alors, on prend 3 fois 5, 15,moins 2 Ca va pas, faut trouver un rapport... On va prendre celui-là...ce serait bien, fois 2 moins 1, tu prends le cinquième, fois 2 moins 1 ça fait neuf... un, deux, trois, quatre, cinq , six ,sept , huit, neuf...

K.C. Pourquoi 3? deux, deux...

A.D. ah oui, t'a raison... moins un

• *est-ce que c'était nécessaire de mettre des étoiles ?*

A.D. K.C. Non...

l'espace n'est pas compris comme un caractère.

A.D. on pouvait mettre un espace ; ça changeait le rapport, aussi; c'est peut-être plus simple...

• *c'était surement plus simple...*

A.D. puisque ça donnait automatiquement le..ce qu'on donnait...

4.3 Troisième phase : résolution de l'exercice 5 (CHIFFRE), et d'un prolongement (reconnaître si un nombre est multiple de 5)

programme produit lors de l'épreuve écrite:

```
10 INPUT Z
20 let A=Mid$(Z,Len(Z)-1,1)
30 Print A
40 END
```

• *bon , on regarde le dernier*

A.D. Stop erreur de type...

notre intervention est sans doute prématurée

• *erreur de type ; ça veut dire que* MID$ *attend une chaîne de caractère*

A.D. ah...donc..

• MID$ *et* LEN. *; il faudrait en fait avoir* Z$ *et* A$...

A.D. on met Z$ et A$

Il s'agit bien d'une incertitude sur la notion de type, et non d'une simple erreur de syntaxe.

K.C. mais non c'est des nombres... je voulais mettre des $, tu m'as dis non... ...

A.D. il nous a donné les dizaines...

même erreur en ligne 20 qu'à l'exercice précédent L'avant dernier chiffre est correctement interprété comme celui des dizaines.

K.C. les dizaines...

• *expliquez moi pourquoi vous avez mis -1.*

A.D. je crois que ça a sonné à ce moment là.

nous mettons en relation l'erreur constatée ici avec la même erreur survenue à l'exercice 4.

• *oui, mais cela vous l'aviez déjà fait, regarde là...vous l'aviez fait ici pour avoir la dernière lettre, et en fait vous aviez vu qu'au lieu de donner le "T" ça avait donné le "U" vous l'aviez corrigé, en fait, vous n'aviez pas bien analysé le truc, là, parce-que ...*

Il semble que dans certains cas, `MID$(X$,I,n)` soit compris comme commençant "après le `I`ème caractère".

A.D. je crois que là, on avait compris, mais que même là, j'ai eu beaucoup de mal, parce que je ne savais pas la dernière ou le dernier(?), après...

• *ah oui...il y une petite confusion peut-être, si je donne une chaîne de caractère, mettons 5 caractère, on ne s'occupe pas de penser que si ça commence là, que si, le premier nombre, mettons, ici ce serait 5, ça commence à celui qui suit...*

K.C. au début c'est ce que je croyais, mais quand j'ai regardé l'exemple, je me suis rendu compte que c'était pas vrai.

Nous proposons un prolongement afin d'examiner si la compréhension de la notion de type a progressé.

• *d'accord, donc en fait, c'était* LEN, *c'était bien sur la dernière lettre que... En fait c'est pas celle qui suit, c'est bien celle qui est pointée... Sauriez-vous faire un programme qui reconnait si un nombre est multiple de 5? Oui, c'est pas des maths... Comment tu sais si un nombre est multiple de 5, s'il est divisible par 5?*

A.D. ben, s'il est divisible par 5, s'il est entier, il est divisible par 5...

• *si je prend 2...*

A.D. divisé par 5, s'il est entier, il est divisible par 5

• *ça donnera des décimales*

A.D. oui mais c'est une entier tout le temps, c'est un entier qui est multiple

• *par exemple, 35 est un multiple de 5, parce-que c'est 7 fois 5...*

K.C. oui, y a pas de virgule, y a pas de décimales...

• *je vais vous donner le moyen ; est-ce que vous savez comment on reconnait si un nombre est multiple de 5?*

A.D. je sais pour multiple de 3 ; je sais pour multiple de 9...

• *par exemple, tu prends 0, 0 c'est multiple de 5, 5 c'est 5 fois 1, 2 fois 5 10, 3 fois 5 15,*

A.D. faut que ça se termine par 0 ou 5 ?

• *c'est ça, le chiffre des unités, c'est soit 0 soit 5, donc vous complétez le programme...*

ajout d'une ligne :
`IF A$=0 THEN PRINT....`

....

A.D. ça va marcher...

nouvelle confusion de type

K.C. ça marche pas , je te l'avais dit

• *vous avez encore une erreur de type...cette fois-ci vous pouvez la trouver.*

La remarque de **K.C.** est exacte ; mais la question de comparer un chiffre et une chaîne de caractère ne présente pas pour les élèves de solution évidente.

K.C. je te l'avais dit ; t'avais mis un truc de chaîne avec des chiffres

A.D. ben corrige

• *pourquoi ça marche pas?*

A.D. parce que ça, ça correspond à des caractère et là c'est un chiffre qu'on avait mis. …

Nous tentons de rappeler aux élèves la nécessité syntaxique des guillemets pour les constantes chaînes, mais sans succès !

• *on cherche à savoir si* A$ *est bien égal (à 0), mais ce n'est pas égal au nombre 0, mais au caractère 0 ; le caractère 0… si on veut savoir si* A$ *est le caractère B, par exemple, tu l'écrirais : IF A$=B ; comment on écrirais le caractère B*

…

Après une tentative infructueuse, nous proposons une situation où les élèves ne peuvent pas échapper aux guillemets; en effet, l'oubli des guillemets dans un affichage est une erreur courante à laquelle les élèves ont certainement été confrontés. Par ailleurs les élèves savent que l'étoile ne peut être un identificateur de variable, et donc ne peut se rencontrer sans guillemets. Cette proposition relève donc surtout de la maïeutique. Il faudrait plutôt reprendre avec les élèves la notion de constante.

• *fais moi en mode direct une ligne qui affiche une étoile*

A.D. PRINT une étoile

K.C. propose la réponse attendue, la maïeutique a réussi, mais ses remarques suivantes montrent qu'elle n'est pas du tout convaincue.

K.C. entre guillemets ! égal entre guillemets zéro

A.D. c'est ça?

K.C. mais non faut pas qu'il soit égal à zéro ! C'est pas égal à zéro

A.D. IF A$=zéro THEN PRINT ; mais non ça va pas si tu le marques comme PRINT…

K.C. je crois pas que ça va marcher…

A.D. moi non plus…

K.C. ça va pas marcher

…exécution…succès…surprise.

A.D. hein !…

une dernière tentative pour mettre les élèves devant les difficultés de la dualité chaîne de chiffres-entité numérique.

• *supposez qu'on modifie votre programme, et qu'au lieu de mettre "0", on mette "00", comme ça?*

A.D. ah oui, alors là, ça va pas marcher!

• *pourtant c'est toujours 0, 00*

K.C. ben oui, mais...

A.D. je sais pas moi; on va voir!

K.C. mais non, ça va pas marcher!

• *d'accord, mais il faudrait plutôt chercher pourquoi ça va pas marcher.*

Il reste cependant une confusion: la sous-chaîne contient seulement le dernier caractère et ne peut donc être égale à "00"

A.D. parce que il faudrait qu'il y ait deux zéro à la fin pour que ça marche.

• *même...*

K.C. même, même parce que là, il écrit un zéro, pas deux zéros

• *il nous donne un seul caractère , de toute façon il ne pourra jamais être égal à une chaîne de deux caractère Et là, si j'écris 35, ça va marcher?*

Seul le test `A$="0"` a été réalisé.

A.D. K.C. ben non, parce que on l'a pas encore fait...

A.D. et après, je suppose que vous allez nous demander le résultat de la division..

• *non, non je n'y pensais pas.*

5. L'installation d'un S.R.T. relatif au traitement des chaînes dans un S.R.T. opératoire plus général (Entretien avec Arnaud P.)

Arnaud P. (**A.P.** dans la suite) était absent lors de la passation de l'épreuve. Bien qu'un peu brouillon, c'est un élève qui réussit en informatique. Il nous a semblé intéressant de regarder comment il réagit à l'exercice de la question 4, plus précisément de savoir s'il réussirait à cette question, ou s'il rencontre les mêmes difficultés que ses camarades. L'entretien a duré une vingtaine de minutes.

• *tu te rappelles ce qu'est* MID$ *?*

A.P. oui, oui ça prend les… à partir de X$, ben X$ c'est la chaine de caractères en question , a c'est à partir de quel nombre, on le prend, et b c'est le nombre qu'on en prend.

• *bien, c'est tout ce qu'on a besoin de savoir.*

A.P. je fais à partir de n'importe quel mot?

• *oui, à partir d'un mot entré au clavier*

programme réalisé :

```
30 PRINT MID$(A$,1,1);
   MID$(A$,2,A$-1);
```

…

• *qu'est ce que tu veux dire, quand tu mets* A$-1 *?*

Confusion entre longueur et caractère

A.P. c'est à dire, c'est la fin de la chaine - 1

• *c'est un nombre qu'il faudrait mettre là ; là* A$ *est une chaine, et 1 est un nombre…ça n'a pas de sens de soustraire un nombre d'une chaine…*

A.P. ben, comment je vais faire, moi ? on avait fait comme cela…

• *faut considérer la longueur, tu as vu la fonction* LEN*, là…*

A.P. LEN, oui, faut que je mette LEN(A$)-1?

• *par exemple, ou on aurait pu donner* LEN *à une …on aurait pu l'affecter à une variable…*

Mise en relation de la décomposition à l'aide de l'affectation et de la notion de sous-programmes (procèdent de la même idée de modularité)

A.P. oui, faire un sous-programme…

• *ce n'est pas exactement un sous-programme…*

…

• il y a quand même une erreur de syntaxe : le A$ doit être entre parenthèses.

L'idée d'utiliser l'affectation n'a pas été retenue.

programme réalisé :

```
30 PRINT MID$(A$,1,1);
   MID$(A$,2,LEN(A$)-1);
40 PRINT MID$(A$,LEN(A$)-1,1)
```

A.P. là il aurait du me mettre TALUS. (résultat :SALUTU) j'ai compris, j'ai tout interverti; ça , ça doit passer là...

... pistage par suppression des points-virgules, ce qui permet que chaque partie soit affichée sur une ligne.

Interversion des lignes; nouvel essai résultat: UALUTS

Les essais en machine se révèlent beaucoup plus productifs que chez les autres élèves.

A.P. c'est la ligne 40 qui déconne (ligne PRINT MID$(A$,LEN(A$)-1,1)) j'enlève le -1 ??

• *qu'est-ce qu'on voulait avoir ?*

A.P. SALUT, TALUS!

Même erreur que **A.D.** et **K.C.**, mais beaucoup plus vite corrigée.

• *la troisième impression, elle correspond à quelle lettre ?*

A.P. à la dernière...

• *ben non justement...*

Nous n'avons pas rencontré chez les élèves précédents ce calcul spontané d'un argument.

A.P. ah non, c'est la première oui! oui! j'ai dû me tromper... après, il me met ALUT, parce qu'il m'écrit tout ! voilà, c'est la ligne 30 qui ne va pas ; parce qu'il me prend tout! là, là...5-1 : quatre...

• *ben, tu prends 4 lettres à partir de la deuxième...*

A.P. ce que je veux, c'est qu'il fasse, c'est A$ - 1

• *regarde l'exemple; t'es d'accord que* LEN, *ça vaut 5, hein...* LEN(A$)-1, *ça vaut 4 ; si je prends 4 lettres à partir de la deuxième..*

De même, rares sont les élèves qui proposent eux-même une valeur de test.

A.P. oui , faut que je prenne -2, je sais pas si ça va marcher ; parce que si je mets autre chose, si je mets quelque chose en six lettres... qu'est ce qu'on peut mettre en 6 lettres? allez, on va mettre TRI POSTAL,... TRI POSTALE; là je dois obtenir ERI POSTALT, normalement; ... ça y est, ça marche!

Annexes au chapitre 7

Ces annexes peuvent être consultées en même temps que le chapitre 7; elles comprennent la description des cours et exercices pendant la période séparant la première épreuve et la seconde épreuve, le texte de cette seconde épreuve, les réponses au premier puis au deuxième problème, et le compte-rendu d'un entretien portant sur la résolution de ces problèmes.

1. Cours et exercices pendant la période séparant la première épreuve (15 janvier) et la seconde épreuve (14 mai)

Entre la première épreuve (rapportée au chapitre **5**), et la seconde épreuve (rapportée au chapitre **6**), les élèves de la classe observée ont eu 12 semaines d'enseignement (un Cours de 1h sans ordinateurs et une séance de Travaux pratiques avec ordinateurs de 1h30 chaque semaine). Nous décrivons dans cette annexe l'enseignement reçu, séance par séance: dans la colonne gauche ci-dessous, on trouvera le résumé du cours, établi à partir du cahier d'une élève , et dans la colonne de droite, en regard, les exercices de programmation, préparés éventuellement en cours, et passés en machine pendant les périodes de Travaux pratiques. Certaines semaines, nous avons pu observer ces séances de travaux pratiques, et nous donnons donc quelques commentaires sur l'activité des élèves.

Cours:

Chapitre "Pour faire des répétitions:"

Les 4 exemples suivants sont proposés:

1: "Le programme *facture*: un programme construit antérieurement a pour fonction d'établir une facture. On demande de l'inclure dans une boucle de façon qu'il se répète jusqu'à ce que l'utilisateur choisisse d'arrêter." *"on ne connaît pas le nombre de répétitions"*.

2: "l'ordinateur demande le nombre de joueurs, chaque joueur joue, puis l'ordinateur dit qui a gagné." *"je connais le nombre de répétitions."*

3: "faire un programme qui écrit 10 fois "**Bonjour**"; *"on connaît le nombre de répétitions"*

4: conjugaison "l'utilisateur choisit:
-1 Présent -2 Futur
on repose la question jusqu'à ce qu'il mette 1 ou 2; *"on ne connaît pas le nombre de répétitions"*.

Programmes entrés en machine en Travaux Pratiques :

```
10 Rem essai sur les boucles
20 print "Combien de
répétitions": INPUT NBtours
30 For compteur=1 to NBtours
40 Print tab(4) Compteur;
"Bonjour"
50 NEXT Compteur
60 Print "Je suis sorti de la
boucle"
70 Print "Compteur vaut ";
Compteur
80 End
```

Commentaire: ce programme constitue une application directe et une illustration du cours; les élèves ont souhaité le voir s'exécuter pour de grandes valeurs de NBtours.

Le programme suivant a été demandé sous forme de problème: *"Pour tous les nombres de 1 à N (choisi par l'utilisateur)*

Les exemples conduisent à un sous-chapitre:
"programmation d'une boucle dont on connaît le nombre de tours"

L'exemple "écrire 10 fois "**Bonjour**"" est repris, et une écriture de programme est donnée en LOGO (langage que les élèves n'ont pas utilisé sur machine): `Repete 10 [ecris "Bonjour"]` Une écriture de programme en **BASIC** est donnée en regard, ainsi qu'une exécution du programme, mais portant seulement sur 3 itérations.L'instruction `FOR...TO...NEXT` est introduite; l'identificateur de la variable de boucle est `COMPTEUR`, identificateur que l'on retrouvera très généralement dans la résolution des exercices. L'exemple d'exécution détaille les valeurs prises par la variable `COMPTEUR`, et les tests effectués par l'interpréteur sur cette variable.

Chapitre "le codage de l'information" Les termes "bit" "octets", sont introduits, à partir d'une explication technologique: "la seule sensibilité de l'ordinateur consiste à savoir détecter la présence ou l'absence de tension électrique dans ses circuits". Un tableau est construit, permettant de connaître le "nombre d'informations différentes" en fonction du "nombre de bits"

Chapitre "introduction aux boucles dont on ne connaît pas le nombre de tours".

Problème posé: *"l'ordinateur demande 2 nombres puis calcule le produit de ces deux nombres (sans l'afficher). Ensuite il demande le résultat jusqu'à ce qu'on le trouve."*

La solution est donnée en pseudo-code sous deux versions: l'une utilise "aller en" et des numéros de ligne:

```
1 afficher "Donner un nombre"
saisir le nombre.
2 afficher "un autre" saisir le
  2ème nombre
```

donner le carré, le cube, l'inverse."
programme:

```
10 CLS
20 Print "Donnez un nombre ":
INPUT NBnombre
30 Print "nbre"; "carré";
"cube"; "inverse"
50 FOR compteur = 1 to NBnombre
60 Print compteur;:Print tab(4)
compteur*compteur;:Print
tab(10)
compteur*compteur*compteur;:
Print tab(16) 1/compteur
70 Next compteur
80 END
```

Ce problème a été résolu par une bonne moitié des élèves; les autres élèves se sont trouvés en échec, et ont recopié une solution écrite au tableau par le professeur. Les difficultés ayant trait à l'affichage des résultats ont interagit avec les difficultés liées à l'itération.

Le problème suivant demande un programme qui *affiche successivement les nombres de 15 à 20;* trois solutions sont données (compteur allant de 15 à 20; compteur allant de 1 à 6: compteur allant de 14 à 10).

Par la suite, le professeur demande un programme pour faire la somme de deux nombres entrés au clavier par l'utilisateur, puis de trois nombres. Puis, il demande que le *nombre d'entrées puisse être choisi par l'utilisateur.* Le but de l'activité est de dégager la nécessité d'une structure itérative, et celle, nouvelle pour les élèves, d'une variable "*cumulative*", c'est à dire d'un variable initialisée à 0 avant le début de l'itération, et évoluant dans le corps de la boucle par une instruction `LET`

```
3 calculer le produit
 4 afficher "
 5 afficher "calcule le
 produit"
 6 saisir la réponse
 7 si réponse différente de
 produit,   aller en 4
 8 afficher "tu as trouvé".
```

L'autre version utilise le couple d'avertisseurs "tant-que...refaire", sans numéros de ligne:

```
afficher "Donner un nombre"
 saisir le nombre.
 afficher "un autre" saisir
 le  2ème nombre
 calculer le produit
 afficher "
 afficher "calcule le
 produit"   saisir la réponse
 tant que la réponse est
 différente du produit
    afficher "faux, donne le
    produit"
    saisir la réponse
    refaire
 afficher "tu as trouvé"
```

Par la suite, l'instruction WHILE...WEND [17] est introduite, et le programme est codé en **BASIC**, puis la correspondance entre les deux versions est donnée de manière générale, sous la forme d'un sous-chapitre: "Comment programmer le WHILE quand on ne l'a pas".

Fonction hasard: "il existe une fonction hasard qui renvoie un nombre décimal compris entre 0 (inclus) et 1 (exclus)".Des méthodes sont données pour obtenir un nombre au hasard entre a et b, un nombre entier au hasard entre a et b.

Du problème à l'algorithme: Une méthode de construction de l'algorithme est décrite:
"a) est-ce que j'ai compris ?
b) est-ce que je sais le faire à la main?

S=S+N. Plutôt qu'une situation problème , l'activité est une recherche guidée dans laquelle l'apport du professeur est dominant.

L'activité trouve un prolongement dans la recherche d'un programme faisant la moyenne de N nombres entrés au clavier, N étant lui aussi entré par l'utilisateur.

Le cours sur le *codage des caractères* a donné lieu aux exercices sur machine suivants:
1- affichage de l'alphabet à l'écran (utilisation de la fonction CHR$).
2- conversion de l'écriture décimale d'un nombre entier positif inférieur à 255 dans son écriture binaire.
3- conversion inverse.
4- utilisation de l'instruction DEF GR$ pour construire un motif dans une matrice de 8x8 points; un GR$ est défini par une suite de 8 octets, l'écriture binaire de chaque octet étant représentative de la ligne correspondante du motif.

Nous avons observé et enregistré au magnétophone la recherche de l'exercice 3 (*conversion binaire-décimale*) par un groupe d'élèves. Cette observation montre que si le principe de la conversion est bien compris par ces élèves, l'articulation des différentes instructions à l'intérieur de la boucle présente une difficulté importante. Trois variables évoluent ici dans la boucle (l'une représentant **le chiffre binaire courant**, l'autre représentant **la contribution au résultat** dans le cas où ce chiffre est 1, la troisième est la **cumulation** des contributions des 8 chiffres binaires). Les élèves font évoluer relativement facilement chacune des trois variables individuellement dans une boucle,

[17] qui correspond à une forme particulière d'itération (forme TANT QUE), où l'alternative de sortie unique est en tête du corps de l'itération ;

c) puisque je sais le faire, j'analyse le problème . Comment est-ce-que je m'y prend pour trouver ? Résultat=algorithme.

L'exemple suivant est étudié:

"le 1er du mois est un lundi; trouver le jour correspondant à une quinzaine donnée; exemple: le 18 est un jeudi" et la solution donnée est la suivante:

"Algorithme :
- on cherche combien de fois il y a 7 dans 18
- on cherche le reste.
- suivant la valeur du reste:
  1 → lundi
  2 → mardi
  3 → mercredi
  4 → jeudi
  5 → vendredi
  6 → samedi
  0 → dimanche"

mais échouent quand il s'agit d'articuler les évolutions des trois variables dans la même boucle. Le calcul du chiffre binaire courant se fait à l'aide de la fonction MID$, ce qui n'est pas sans poser problème.

La séance de travaux pratiques suivant le cours sur *"Comment programmer une boucle..."* donne lieu au codage en machine du programme écrit sous forme de pseudo-code en cours, puis les élèves ont à résoudre le problème du nombre d'occurrences d'un caractère donné dans une chaîne. Les deux problème suivants sont ensuite proposés aux élèves:

- faire un programme qui *simule le lancement d'une pièce de monnaie.*
- *l'ordinateur choisit un nombre entier au hasard entre 1 et 100; on doit le trouver, l'ordinateur nous guide en disant: trop grand ou trop petit.*

2. Le texte de l'épreuve (EPR2)

Exercice 1: (problème CLASSE)

Dans une classe de 12 élèves, un devoir a été noté de 0 à 20; le professeur décide de faire un programme lui permettant, en entrant le nom d'un des élèves, d'obtenir en sortie sa note au devoir.

Son programme utilise une chaîne de caractères rangée dans la variable CLASSE$ et comportant successivement le nom de chaque élèves, suivi de la note qu'il (ou elle) a obtenue au devoir (indiquée avec 2 chiffres)

```
10 LET CLASSE$
="DUPOND09ALFRED12ARTHUR14LAMBERT11MARIE05VICTOR12
DUPONT16DUVAL14CHARLIE19ARMAND13DUBOIS15DUMONT14"
20 PRINT "Donnez le nom de l'élève "
30 INPUT NOM$ 40 LET LC=LEN(CLASSE$)
50 LET LN=LEN(NOM$)
60 FOR I = 1 TO LC
70 IF NOM$=MID$(CLASSE$,I,...) THEN LET
NOTE$=MID$(CLASSE$,...,...)
80 NEXT I 90 PRINT "La note est ";NOTE$
```

voir chapitre 4 §1.4.2

Tu dois compléter les trois emplacement libres (marqués par les "...") de façon à ce que le programme donne bien la note de l'élève.
Ecris dans ta réponse la ligne 70 complétée, et explique.

Exercice 2: (problème **CRYPTAGE**)
Deux correspondants souhaitent échanger des messages confidentiels.
C'est pourquoi ils conviennent du cryptage suivant de leurs message:

- ils conviennent d'une chaîne de caractères comportant les 26 lettres de l'alphabet dans un ordre donné; on prendra ici la chaîne "azertyuiopqsdfghjklmwxcvbn"
- le cryptage consiste à remplacer chaque lettre du message par la lettre obtenue comme suit: s'il s'agit de la dernière lettre de la chaîne (ici "n"), on la remplace par la première de la chaîne (ici "a"), sinon, on la remplace par la lettre qui la suit dans la chaîne. Ainsi, le message "bonjour" est crypté en "npakpit".

On veut un programme tel que:

- en entrée on donne une lettre de l'alphabet,
- en sortie on obtient la lettre correspondante dans le cryptage.

Ecris l'algorithme et le programme BASIC qui résolvent ce problème .

3. Les réponses au premier problème (CLASSE)

Les trois tableaux ci-dessous donnent élève par élève les réponses aux trois écritures demandées suivies de l'explication fournie éventuellement par l'élève:

3.1. Longueur de la sous-chaîne à laquelle comparer NOM$

Caroline C.	LN	...
Benoît G.	LN	cite une définition concernant MID$
Florence P.	LC	"dans la parenthèse, on met LN qui est noms car le programme affichera la classe, le nom".
Stéphanie G.	LN	"si le nom=(2ème nom, des douze élèves, est Alfred..."
Stéphanie Bé.	LN	"dans le programme à la ligne 50 LET LN$=LEN(NOM$) alors dans (CLASSE$,I,...: sera remplacé par LN."
Stéphanie Be.	LN	"j'ai trouvé ça sur une feuille, mais j'en sais rien".
Pierre P.	LN	
Florent R.	LN	cite une définition concernant MID$
Karine C.	LN	"on prend la chaîne que l'on considère CLASSE$, et l'on prend la lettre égale à I..."
Armelle D.	LN	"si le nom de l'élève correspond, dans la chaîne de caractères, au même nombre de lettre,..."
Pierre E.	LN	"nombre de lettres qu'il faut afficher c a d longueur du nom"
Fréderic P.	LN	"LN parce-que, pour que l'ordinateur "reconnaisse" le nom de l'élève, il faut que le nom entré au clavier ait le même nombre de lettres que le nom de l'élève."

Arnaud P.	LN	"je donne la variable "LN" au microprocesseur comme nombre de caractères à étudier pour qu'il puisse reconnaître le nom en comparaison avec nom$"
Alexandra J.	LN	"Comparer le nom de l'élève tapé au clavier(...) à la chaîne de caractères CLASSE$..."

3.2. Position à partir de laquelle rechercher la note comme sous-chaîne de CLASSE$

Caroline C.	LN + 1	"on a LN+1 parce qu'il faut prendre le nom + 1 caractère pour avoir le début de la note..."
Benoît G.	I + LN...	
Florence P.	LN	...
Stéphanie G.	I	"si le nom=(2ème nom, des douze élèves, est Alfred) alors la note=(2ème note, des douze élèves, est douze)"
Stéphanie Bé.	I	
Stéphanie Be.	LN1	
Pierre P.	LN + 1	"on prend le nom + 1 lettre, c'est à dire le premier chiffre de la note"
Florent R.	I + LN	...
Karine C.	LN	"on prend la chaîne que l'on considère CLASSE$ et l'on prend la lettre égale à I; à partir de I, on prend la longueur du NOM$, ..."
Armelle D.	LN	"... alors la note est prise dans la chaîne de caractères classe$, après la longueur du nom..."
Pierre E.	LN + 1	"La note suit le nom, donc il faut commencer à afficher après le nom."
Fréderic Portier	LN + 1	"Pour sortir de CLASSE$ la note de l'élève, l'ordinateur doit prendre le premier caractère de la note, c'est-à-dire le caractère suivant le dernier caractère du nom de l'élève..."
Arnaud P.	LN	"Je donne LN comme caractère de départ pour que le microprocesseur analyse la note qui se trouve derrière le nom..."
Alexandra J.	LC - LN + 2	"quand on a trouvé ce nom dans la chaîne de caractères CLASSE$, alors prendre dans cette chaîne les 2 caractères qui suivent le nom"

3.3. Longueur de la sous-chaîne à calculer pour obtenir NOTE$

Caroline C.	2	"...et prendre 2 caractères pour avoir la note qui comporte 2 caractères."
Benoît G.	2	...
Florence P.	2	...
Stéphanie G.	LN	
Stéphanie Bé.	LN	

Stéphanie Be.	...	
Pierre P.	2	"à partir de là, on prend deux lettres, soit la note puisqu'elle est notée sur 20, donc 2 chiffres seulement."
Florent R.	2	"2 c'est la longueur de la chaîne de caractères à chercher"
Karine C.	2	"... et pour obtenir la note, on prend deux caractères de plus."
Armelle D.	2	"...après la longueur du nom, on prend deux caractères."
Pierre E.	2	"Il faut prendre 2 caractères."
Fréderic P.	2	"... et il doit prendre 2 caractères."
Arnaud P.	2	"... et 2 car la note comporte 2 chiffres."
Alexandra J.	2	"... les deux caractères qui suivent le nom."

4. Les réponses au second problème (CRYPTAGE)

nous étudions successivement:

1. l'existence d'une recherche de la position de la lettre donnée dans la chaîne "azert..." , et dans le cas où cette phase de recherche existe, la façon dont elle est construite.
2. le calcul de la lettre résultat à partir de la position trouvée pour la lettre donnée (ou autrement).
3. la prise en compte du cas particulier constitué par la lettre "n".

4.1. Recherche de la position du caractère entré par l'utilisateur dans la chaîne "azertyuiopqsdfghjklmwxcvbn": existence et construction

Certains élèves ne considèrent pas cette étape de la programmation; du fait de leur conception des fonctions sur les chaînes, elle n'est pas pour eux, nécessaire; nous indiquerons pour ces élèves "pas de recherche".

	expression dans l'algorithme	expression dans le programme.
Caroline C.	"voir sa place dans..."	pas de programme
Benoît G.	"faire une boucle se répétant LA fois; si L$ est égal à la chaîne de 1 caractère , et à partir du Xème caractère ..."	FOR X = 1 TO LA IF L$=MID$(ALPH$,X,1)... NEXT X
Florence P.	pas de recherche	pas de programme
Stéphanie G.	"il la place dans sa chaîne de caractère , la positionne par . rapport au compteur..."	FOR I=1 TO 26 LET MID$=(C$,L$,I+1) (pas de NEXT I)
Stéphanie Bé.	"il la place dans sa chaîne de caractère , la positionne par . rapport au compteur..."	FOR I=1 TO 26 LET MID$=(C$,L$,I+1) (pas de NEXT I)
Stéphanie Be.	pas de recherche	pas de programme

Pierre P.	pas de recherche	programme ne comportant pas de boucle
Florent R.	13 lignes constituées comme: "si X$="n"... ..... "si X$="f"...	pas de programme
Karine C.	pas de recherche	programme ne comportant pas de boucle
Armelle D.	pas de recherche	programme ne comportant pas de boucle
Pierre E.	pas de recherche	programme ne comportant pas de boucle
Fréderic P.	13 lignes constituées comme: "si AB$="A" .... "si AB$="D"	13 lignes constituées comme: IF AB$="A"... ... IF AB$="D" ...
Arnaud P.	"La lettre du message non codé est recherchée dans la chaîne de caractères.."	FOR I=1 TO 26 IF L$=MID$(CRYPT$,I,1) NEXT I
Alexandra J.	"la comparer à toutes les lettres de la chaîne de caractères."	pas de programme.

4.2. Expression du caractère résultat en fonction du caractère donné ou de sa position

	dans l'algorithme	dans le programme
Caroline C	"écrire la lettre qui la suit "	pas de programme
Benoît G.	"C$ est égal au caractère à sa droite..."	LET C$=MID$(ALPH$,X+1,1)
Florence P.	"prendre la lettre qui la suit "	pas de programme
Stéphanie G.	"en ajoutant 1 pour faire la lettre demandée."	PRINT L$+1
Stéphanie Bé.	"en ajoutant 1 pour faire la lettre demandée."	PRINT L$+1
Stéphanie Be.	"cette lettre se trouve après la lettre choisie"	pas de programme .
Pierre P.	"écrire la lettre qui la suit "	LET NL=MID$("L",L+1,1)
Florent R.	...alors Y$="a" ...alors Y$="d"	pas de programme ...
Karine C.	"on prend la lettre qui suit "	C$=MID$(A$,C$+1,1)
Armelle D.	"prendre la lettre qui la suit "	X=MID$(A$,X+1,1)
Pierre E.	"la remplacer par la lettre correspondante qui se trouve après la lettre X (X+1)."	LET NOUVELLE$= MID$(CHAINE$,LETTREALPHA$+1,1)
Fréderic P.	...alors Z$="N" ..alors Z$="F"	...THEN Z$="N"...THEN Z$="F"
Arnaud P.	"il cherche quelle est la lettre qui la suit dans CRYPT$ "	LET C$=MID$(CRYPT$, I+1,1)
Alexandra J.	"quand on l'a trouvé alors prendre la lettre juste avant la lettre demandée "	pas de programme

4.3. Prise en compte du cas particulier constitué par la lettre "n"

	dans l'algorithme	**dans le programme**
Caroline C.	en fin d'algorithme "(si c'est n écrire a)"	pas de programme
Benoît G.	en début de corps de boucle: "si x=LA alors C$ est égal au premier caractère de la chaîne"	IF X=LA THEN C$=MID$(ALPH$,1,1)
Florence P.	début d'algorithme: "si c'est a alors n sinon..."	40 IF
Stéphanie G.	pas de prise en compte	pas de programme

Stéphanie Bé.	pas de prise en compte	pas de prise en compte
Stéphanie Be.	pas de prise en compte	pas de prise en compte
Pierre P.	"si la lettre est n écrire a sinon..	fin du programme: `IF L=n THEN print a else print "NL"`
Florent R.	pas de prise en compte particulière (du fait de la structure en 13 instructions conditionnelles)	
Karine C.	en fin d'algorithme "mais si la lettre donnée correspond à la dernière de la chaîne, alors on prend la première de la chaîne"	IF C$=MID$(A$,LEN(A$)-1,1) THEN PRINT "a" ELSE ...
Armelle D.	en début de traitement "si la lettre est la dernière de la chaîne, prendre la première, sinon..."	IF X=MID$(A$,LEN(A$)-1,1) THEN PRINT "A" ELSE ...
Pierre E.	pas de prise en compte	pas de prise en compte
Fréderic P.	pas de prise en compte particulière (du fait de la structure en 13 instructions conditionnelles)	
Arnaud P.	pas de prise en compte	pas de prise en compte
Alexandra J.	pas de prise en compte	pas de programme

5. Confirmation de l'interaction Objets-traitements (Compte-rendu d'un entretien suite à la passation de l'épreuve 2.)

L'entretien s'est déroulé sur une durée de 1h30, le 19 mai 1987. Contrairement à la première épreuve, nous avons pu, avec les réponses à l'épreuve sur papier, tracer les grandes lignes d'une analyse des difficultés des élèves. C'est pourquoi nous nous sommes limité à un entretien, travaillant avec trois élèves rencontrant des difficultés variées, de façon à obtenir une confirmation de notre analyse: deux de ces élèves (Florence P. et Alexandra J.) rencontrent lors de la première épreuve (chapitre 5) les difficultés *D.Langage.* et *D.Sem.Cha.* et elles obtiennent un indice de réussite nul, et la troisième (Armelle D.) intègre mieux les éléments généraux du langage et les chaînes, et obtient un indice de réussite de 3 sur 5, sa difficulté dominante étant *D.Ord.Card.*.(voir tableau § 4.4 du chapitre 5). La procédure est la même que pour les entretiens faisant suite à la première épreuve: les élèves sont devant l'ordinateur, avec le résultat de leur recherche écrite ; nous enregistrons au magnétophone, prenons des notes et relançons la recherche. Dans le compte-rendu ci-dessous, le dialogue enregistré est dans la colonne de droite, les commentaires sont dans la colonne de gauche. Nos interventions sont en italiques. Les remarques des élèves,sont précédées de leurs initiales (Florence P.: **F.P.**, Alexandra J.: **A.J.**,. Armelle D.: **A.D.**)

Solution par écrit de **A.J.** :

```
70 IF NOM$=MID$(CLASSE$,I,LN)
                        THEN LET
NOTE$=MID$(CLASSE$,LN-LN+2,2)
```

Solution par écrit de F.P. :

```
70 IF NOM$=MID$(CLASSE$,I,LN)
                        THEN LET
    NOTE$=MID$(CLASSE$,LN,2)
```

Le programme a été préparé par le professeur sur disquette; il suffit de compléter

• *Bon alors, on prend la solution d'Alexandra...*

A.D. Elle copie pas ce qu'elle a copié, c'est n'importe quoi

F.P. oui, alors on essaye...

A.J. on a le droit d'essayer, Armelle !

A.J. dis moi où il est le "moins"

F.P. écoute tu va écrire seulement `LN-2`

• *on essaye la solution d'Alexandra?*

La réponse de **A.J.** a été interprétée dans le dépouillement de la façon suivante: LN-LN+2 est la chaîne CLASSE$ que l'on aurait privé des caractères jusqu'à NOM$ (compris), et où l'on prendrait 2 caractères.

A.D. c'est pas une solution, c'est ce que l'autre avait trouvé c'était stupide... on avait vérifié avec **Karine**

F.P. oui, mais elle veut absolument vérifier

A.D. ça veut rien dire, la longueur de classe moins la longueur du mot...

La réponse de **F.P.** (rencontrée chez 7 élèves sur 13) s'interpréte comme la compréhension de LN en tant que référence ordinale marquant le dernier caractère de NOM$, y compris comme sous-chaîne de CLASSE$.

F.P. tu vas écrire "plus 2", "plus"...mais non, tu as écris EN! tu mets seulement LN, tu effaces LN; non pas là, là!, tu écris LN+2 ... +2.. fermer la parenthèse...

Essai.. entrée DUPOND pas de réponse...

A.J. ça marche pas...

De manière générale, nous avions noté que le statut donné aux ordinaux (en particulier à la fonction longueur) par les élèves, leur permet d'effectuer des opérations arithmétiques sur ces ordinaux, sans qu'ils cessent de représenter intrinsèquement l'objet sur lequel ils portent.

- *essayez de comprendre ce que vous faites; qu'est-ce que ça peut vouloir dire,* LN+2 *ça correspond à quoi?*

A.J. la longueur plus 2

A.D. c'est dans le vide, y a rien !

A.J. alors c'est LN ?

F.P. alors il nous faut ?

A.J. LN, on va bien voir..

On constate ici que la réponse de **A.J.** ne peut être considérée comme une étourderie ou une réponse "au hasard", puisqu'elle maintient sa volonté de la tester malgré la contestation très vive de **A.D.**, et bien qu'elle ne soit pas capable de la justifier par un raisonnement.

F.P. il nous faut la longueur du mot plus 2...

A.J on va bien voir

F.P. en fait ce qu'il nous faut, c'est la longueur du nom..

essai avec LN: entrée LAMBERT résultat AL

La réponse AL de l'ordinateur est interprétée de façon sémantique (AL, pourrait, dans le contexte,être le prénom d'un élève), et non de façon formelle comme constituée de deux caractères obtenus comme résultat du traitement spécifié par les élèves dans leur programme.

A.J. on a déjà avancé hein?

F.P. mais normalement, il doit pas nous donner le prénom, la-dessus?

A.J. faut que tu rentres le prénom...

F.P. oui, mais le prénom, il est pas dans la liste...

même après un an de pratique, la représentation de l'ordinateur reste éloignée de celle d'un simple dispositif automatique, puisque **F.P.** ne serait pas étonnée que l'ordinateur retourne des prénoms qui n'ont jamais été rentrés comme données.

A.J. et ce que tu avais marqué c'est pas ça?

F.P. non c'est pas ça... ... Non, ça c'est bon LN !

• *est-ce que vous pouvez trouver pourquoi on obtient "AL"; expliquez moi comment ça se passe dans la chaîne? à quoi est égal* LN *?*

La *longueur*, c'est la chaîne ! et on peut faire une addition. L'entretien confirme les conclusions du dépouillement.

F.P. LN est égal au nom demandé, c'est à dire LAMBERT ... plus 2 lettres.

• *Attends,* LN *c'est un nom ou ?*

F.P. c'est une chaîne...? c'est une chaîne?

• *qu'est ce qu'il y a à la ligne 50*

F.P. LET LN=LEN...

• *c'est une chaîne ?*

La *longueur*, c'est toute la chaîne!

F.P. ben la longueur, c'est à dire tout le nom demandé, quoi?

• *la longueur, c'est le nombre de caractères...*

F.P. ben oui!

• *c'est pas tout le nom demandé, si je te dis* LAMBERT, *et si je te dis 7, est ce que c'est la même chose ?*

F.P. ben non, c'est pas la même longueur, c'est à dire ça a pas le même nombre...

• *7 est la longueur de* LAMBERT*!*

F.P. ben oui...

• *alors, ici,* LN *vaut ?*

F.P. ben il vaut 7..

• *d'accord, alors* LN + 2 ?

longueur peut "valoir" 7; elle n'en est pas moins représentative de la chaîne dont elle est la longueur.

F.P. ben LAMBERT plus 2 caractères!

• *non!*

F.P. ah ça va faire longueur + 2 longueurs, c'est à dire 2 nombres...

• *c'est-à-dire 9 attendez, d'où est-ce qu'il sort ce* AL*?*

le programme a été interprété selon une signification globale supposée par l'élève, et non selon les régles du langage.

F.P. justement, je vois pas du tout...

• *ben où est-ce qu'on a* AL *dans la chaîne ?*

F.P. ALFRED

• *on a bien deux caractères, là; il s'est pas trompé. Pourquoi on a* ALFRED *?*

A.J. une, deux, c'est la deuxième lettre d'ALFRED

F.P. une deux trois...ça a pris en fait en fait une première longueur, la longueur de LAMBERT c'est à dire 7, plus 2 caractères, et ça a pris la deuxième lettre

• *ben voilà... où est ce qu'on voulait la trouver cette note ?*

cohérent avec l'interprétation donnée dans le dépouillement.

F.P. en fait nous, il nous la faut juste après la longueur LAMBERT, la longueur bien précise LAMBERT.

• LAMBERT,*il est là...si on commence sur le* 1 *ici, comment on peut le calculer..*

F.P. ben la longueur LAMBERT, moins toutes les longueurs qu'il y a avant, plus 2 caractères...

• *oui... enfin, moins toutes les longueurs qu'il y avant???*

A.J. non, on ajoute

• *on ajoute toutes les longueurs qu'il y avant ! vous voyez là...*

LN est ensuite remplacé par LN + I suite à une intervention de **A.D.**

essai réussi sur deux noms: LAMBERT et MARIE

• *alors, vous m'expliquez comment ça marche..*

F.P. bon alors, la longueur du nom choisi plus I qui est... euh...

A.J. I c'est... attend

Le statut de I n'est pas clair, même s'il a été reconnu comme compteur.

F.P. I je l'ai ici (coup d'oeil au cahier)... plus le compteur...c'est à dire euh... à partir du premier caractère jusqu'à la fin de la classe... à la fin de la longueur de la classe...

• I *vaut* LN *???*

F.P. ben non!

A.J. I est égal à 1!

F.P. I c'est le nom de ...

A.J. de chaque élève ...?

Il est probable que, pour **A.J.** , comme pour d'autres élèves (voir dépouillement), I constitue un *index sur les noms*, et non sur les lettres de la chaîne.

F.P. non c'est le nom de la courbe... c'est le numéro du compteur, comme quand on fait FOR C=...

• *d'accord, ça prend toutes les valeurs de* 1 *à* LN... *pourquoi est ce que vous faites* LN+I *??. si Armelle ne vous avait pas dit d'écrire* LN+I , *qu'est ce que vous aviez envie d'essayer?*

A.J. on aurait tout essayé jusqu'à ce que ça marche!

F.P. je n'aurais pas pensé à écrire ça!

A.J. moi non plus... pourquoi I ??

F.P. en fait on a pris ...

• *demandez à Armelle de vous expliquer pourquoi elle a fait* LN+I

A.J. Armelle pourquoi tu as fait LN+I ?

A.D. vous êtes nuls, vous êtes nuls...

• *ce n'est pas une explication! ... elle ont dit: on n'aurait pas pensé à* I ...

F.P. parce que Pierre te l'as dit ?

La compréhension de **A.D.** reste très superficielle, même si des essais l'ont persuadé de la validité de son programme.

A.D. non pas du tout, nous on a pensé à I parce que on s'est dit que c'était un truc avec le I , qu'il pouvait pas servir à rien! il y a cela FOR I =

• *il fallait faire quelque chose avec le* I *? on pourrait le retirer par exemple..*

A.D. comment cela?

nous tentons d'obtenir une validation du programme en contestant le raisonnement de **A.D.** sur le plan logique, mais sans succès.

• LN-I , *ou pourquoi pas* LN *multiplié par* I *? ... essayez de la faire fonctionner sur un exemple! si je cherche* ALFRED, *par exemple,* I *va être égal à combien?*

de la même façon que **A.J.**, **A.D.** interprète I comme *index sur les noms*, et non sur les caractères de CLASSE$.

A.D. ALFRED est en deuxième position...

• *oui...*

F.P. c'est la longueur de ALFRED plus...

• LN, *c'est la longueur de* ALFRED

A . D. plus ce qu'il y avant... et on prend deux caractères...

• I *c'est égal à ce qui est avant? si on parle de position,* I *c'est égal à quoi...*

pour **A.D.**. `I` est égal à 2; la somme `LN+I` n'est certainement pas comprise comme la somme de deux entiers.

A.D. ALFRED est en deuxième position, il est là, donc...

F.P. le 2, ça veut dire seulement tu prends les deux dernières notes..

A.D. attends, moi j'en suis qu'à "plus `I` ", moi...

• *c'est à dire cette position là...*

A.D. oui..

• *si j'ajoute* `LN`...

F.P. ça va prendre la longueur d'ALFRED, et après, on va prendre les deux caractères qui suit...

A.D. moi, je comprends ça comme cela, mais je ne peux pas l'expliquer...

• *bon on passe à l'exercice suivant...*

---

résolution du second problème (*une chaîne "code" étant donnée, faire correspondre à chaque caractère celui qui le suit dans la chaîne*)

Cette fois, la nécessité d'une itération a été comprise; c'est le signe d'une bonne assimilation du processus de recherche dans la chaîne vu avec le premier problème .

F.P. bon `10 CLS`...faut reprendre le même principe que...on donne une lettre de l'alphabet, il te la ressort cryptée... je vais faire une boucle... je pense qu'il faut faire une boucle

(... dicte à **A.J.** en se guidant sur le programme précédent)

toujours la confusion longueur-chaîne (`A$` au lieu de `len(A$)` à la ligne 30, et `X$` au lieu de `len(X$)` à la ligne 50

```
10 CLS
20 LET
A$="azertyuiopqsdfghjklmwxcvbn"
 30 INPUT "Donnez une lettre";X$
40 FOR I=1 TO A$
50 IF X$=MID$(A$,I,X$) THEN LET
C$=MID$(A$,X$+I,1)
60 NEXT I
70 PRINT"Le cryptage est ";C$
```

essai: erreur `type mismatch in 40`

F.P. je comprends pas ...oui, je vois... faut pas qu'on mette `A$`, et après, faut qu'on mette ici... `LET`...comme ici (dans le programme précédent) met `AA`.

ajout des lignes 31 et 32:

```
31 LET AA=LEN(A$)
32 LET XX=LEN(X$)
```

et remplacement de `A$` par `AA` dans la ligne 40; entrée d'un `a` minuscule; résultat vide

F.P. faut qu'on change aussi ce qu'il y a là (ligne 50) là c'est bon normalement...non, ça ne marche pas.

• *il n'y a pas d'erreur de type, cette fois* `X$` *ici c'est un* `A` *majuscule...*

F.P. c'est ça notre erreur... oui, ça marche,

difficultés linguistiques pour exprimer une relation fonctionnelle.

A.J. non, "a" c'est "n"

F.P. ah, non c'est "n" qui est "a", "a" normalement c'est "z", donc ça marche!

• *essayez d'autres lettres pour voir!*

F.P. tu vas prendre la lettre... tiens, fais "r"

A.J. non justement, je vais faire "n" pour voir si ça fait "a"!

essai avec l'entrée `n`; résultat vide

F.P. oh, ça recommence... ... ... faut qu'on fasse un `IF`...

ajout d'une ligne `80 IF X$=n THEN PRINT a`

F.P. faut qu'on mette entre guillemets

correction essai

A.J. ça marche...

La difficulté à insérer le cas particulier découle de la non-distinction entre l'indication du résultat (ici par l'affectation à une variable) d'une part, et l'affichage de ce résultat d'autre part. L'affectation pose toujours un problème de signification en tant qu'action.

F.P. ça marche, mais ça le donne à la ligne... faut peut-être qu'on change le numéro de ligne c'est tout.. (après nous avoir consulté du regard) non c'est pas tout, bon ,c'est pas tout!

• *le problème c'est que vous faites deux affichages: on a une entrée, faut une seule sortie...faut que la sortie soit* `C$`, *c'est à dire ici (*`ligne 50`*), on le donne à* `C$` *ici, il faut donner le résultat à* `C$` *!*

F.P. faudrait qu'on le fasse juste après `C$` , qu'on l'écrive après `C$` !

A.J. non juste avant !

• *faut qu'on le donne à* `C$` *, le résultat, au lieu de la faire imprimer...*

A.J. c'est en ligne 65...

F.P. je ne comprends pas là... ... Monsieur:si j'écrivais ici: `IF X$` est égal à vide `then Print "A"`, est-ce que ça marcherait?

• `X$` *est égal à ???*

F.P. Non, `C$` est égal à...

• *c'est pareil, tu auras toujours* `PRINT` *le cryptage*

F.P. et à la ligne !

• *ici on a un résultat* `PRINT` *le cryptage est ...* `C$`*... Où on l'a obtenu* `C$`*?*

F.P. juste au dessus...

• *éventuellement, on lui a donné ici (*`ligne 50`*) et là(*`ligne 70`*) c'est juste l'affichage à la fin;il faut, que si tu veux ,le résultat soit obtenu par* `LET C$=` *quelque chose. Où est qu'on pourrait le mettre, ça donc, là le "*`n`*" on l'a pas trouvé, ou on l'a trouvé, et on pris pour* `C$` *la lettre suivante donc , comme il n'y en a pas, c'est vide... alors effectivement, on pourrait mettre* `IF C$=` *vide ,* `THEN` *et si on met* `PRINT` *"*`a`*" on va avoir "*`a`*", puis à la ligne "le cryptage est "; ça n'a pas d'intérêt ce qu'il faut changer, ici c'est* `C$` *faut changer* `C$`*, tu vois pas ?*

F.P. `non`

• *il faut bien au départ que* `C$` *soit égal à vide , pour qu'on arrive ici et faut qu'après il soit égal à...*

F.P. ...

• *il faut qu'il soit égal à "*`a`*"; et comment on peut*

changer la valeur de C$ *?*

Difficulté à utiliser une même variable pour deux valeurs successives différentes.

F.P. ben LET, mais en même temps, faut que ça garde que C$ est égal à ça (fin de la ligne 50)

- *non, non, ça ne nous intéresse plus ! puisque* C$ *était vide, ce qui nous intéresse c'est que* C$ *soit égal à "a"!*

F.P. oui, mais dans les calculs, il sera pas toujours égal à "a"!

- *on dit seulement* IF C$=*...c'est à dire si il a reçu une valeur vide... Si je fais un test après la ligne 60*

F.P. IF C$ = vide...

- THEN ?

F.P. THEN PRINT "a"

- *ben non parce que ça fera pareil, tu auras un "a" puis après, "le cryptage est" tu auras pas changé* C$...

F.P. donc THEN PRINT MID$, na , na, na , mais garder seulement...ah non je ne vois pas!

- *ce qu'il faut changer, c'est la valeur de* C$ *; ce qui est désagréable, ici, c'est que dans le cas de "n", on n'a rien, c'est un cas particulier...C'est le seul cas où l'on n'a rien... Donc on décide que dans le cas où c'est rien, on le transforme en "a"*

F.P. oui, mais je vois pas où l'écrire !

- *de toute façon, avant 70, après 60, faut qu'on ait fait toute la boucle... d'accord?*

F.P. on efface la ligne 80;

- *et on fait une ligne 65*

La difficulté à distinguer affectation et affichage est tenace.

F.P. on écrit:IF C$ est égal à vide THEN PRINT

- *non justement pas* PRINT, *faut qu'on ait une impression seulement en 70, dans ces lignes là faut qu'on donne une valeur à* C$ *; attends, je vais te le dire:* THEN C$ =...

F.P. ah oui C$ ="a", j'ai compris...

Annexes au chapitre 8

1. Le texte de l'épreuve (version BASIC sans numéro de ligne) passée dans la classe F

Nous avons indiqué dans le paragraphe de présentation de l'énoncé de l'épreuve, que la classe F a disposé d'un énoncé en **BASIC**, langage utilisé dans cette classe, en fait très peu différent de l'énoncé en **Pascal** donné dans le chapitre; voici le texte de cet énoncé:

Exercice 1: (**CLASSE**)
Dans une classe de 12 élèves, un devoir a été noté de 0 à 20; le professeur décide de faire un programme lui permettant, en entrant le nom d'un des élèves, d'obtenir en sortie sa note au devoir.
Son programme utilise une chaîne de caractères rangée dans la variable CLASSE$ et comportant successivement le nom de chaque élèves, suivi de la note qu'il (ou elle) a obtenue au devoir (indiquée avec 2 chiffres).

```
    CLASSE$ ="DUPOND09ALFRED12ARTHUR14LAMBERT11MARIE05VICTOR12
           DUPONT16DUVAL14CHARLIE19ARMAND13DUBOIS05DUMONT14"
    PRINT "Donnez le nom de l'élève  "
    INPUT NOM$: REM entrée du nom d'un élève de la classe
    LN=LEN(NOM$): REM LN est la longueur du nom de l'élève
    I=0
    REM  début de la boucle de recherche du nom
    DO   I=I+1
-->        LOOP UNTIL NOM$=MID$(CLASSE$,I,...)
    REM  sortie de la boucle
-->       NOTE$=MID$(CLASSE$,...,...)
    PRINT "La note est ";NOTE$
```

Tu dois compléter les trois emplacements libres (marqués par les "...") de façon à ce que le programme donne bien la note de l'élève.
Ecris dans ta réponse les deux lignes marquées par des flèches, complétées, et explique.

Exercice 2: A) (**BONJROU**) On veut un programme tel que:
L'utilisateur entre une chaîne de caractères: X$, puis un nombre N inférieur à la longueur de X$. L'ordinateur affiche une chaîne R$ formée à partir de X$, mais où la dernière lettre de x$ apparaît avancée en Nième position:

exemple: X$: bonjour
N: 5 R$: bonjrou
N: 1 R$: rbonjou

L'idée est de décomposer x$ en trois parties:
exemple pour X$:bonjour N:4 A$: bon B$: jou C$: r
Complète le programme suivant (lignes marquées par des flèches)

```
   INPUT X$: REM entrée d'une chaine par l'utilisateur
      L=LEN(X$): REM L est la longueur de X$
      INPUT N: REM entrée d'un nombre par l'utilisateur
-->   A$=MID$(X$,....,.....): REM A$ est la première partie de X$.
-->   B$=MID$(X$,....,.....): REM B$ est la seconde partie de X$
```

```
-->     C$=MID$(X$,....,.... ): REM C$ est le dernier caractère de X$
-->     R$= .......
        PRINT R$
```

B) (**RUOJNOB**) On veut cette fois, en utilisant le programme ci-dessus, à partir de la chaîne entrée par l'utilisateur, obtenir la chaîne retournée:
exemple: X$: bonjour R$: ruojnob
Explique comment tu peux utiliser le programme ci-dessus pour obtenir ce résultat; écris le programme complet pour résoudre B.

2. Les réponses à l'épreuve

Chaque élève est repéré par un numéro dont le premier caractère est la lettre qui repère la classe; on trouvera dans les tableaux suivants l'orientation décidée par le conseil de classe pour chacun de ces élèves (information exploitée au paragraphe **5**), puis les réponses aux différentes questions posées dans l'épreuve.

2.1. Orientation des élèves, et réponses à l'exercice 1

L'indice de réussite vaut **0** si la structure de donnée n'est pas comprise, **1** si la structure est comprise, mais qu'il y a des erreurs, en particulier dans le calcul de l'ordinal"position note", **2** si la réponse est correcte).

NUMERO	Orientation	Longueur du nom	position note	Longueur note	indice réussite
B01	1S	LN	I	LN	0
B02	Redouble	LN	NOM	I	0
B03L	Redouble	(note)-1	I	LENGHT(NOM)1,2	0
B04	1S	LN	I+LN	2	2
B05	Redouble	LN	I	Length(Nom)	0
B06	Redouble	LN	NOM	LN	0
B07	Redouble	LN	NOM	I	0
B08	1S	LN	I	NOM	0
B09	1S	LN	LN-2	2	1
B10	Redouble	6	5	2	0
B11	1S	LN	LN	2	1
B12	Redouble	LN	LN	I	0
B13	1S	NOTE	NOM	LN	0
B14	Redouble	1	LN	I	0
B15	1S	LN	I+LN	2	2
B16	1S	LN	NOM	I	0
B17	1S	LN	LN	2	1
B18	1S	LN	I	LN-2	0
B19	1S	LN	I	NOTE	0
B20	1G	LE.(NOM)-1	LE.	(NOM)-12	1
B21	1S	LN	NOM	LN	0
B22	1S	LN	LN	I	0
B23	1G	LN	LN	I	0
B24	1G	LN	LN	I	0
B25	1G	LN	LN	I	0
B26	1B	7	I	2	1
D01	1S	LN	I+LN	2	2

D02	Redouble	LN	I+LN	2	2
D03	1S	LN	I	LN+2	0
D04	1S	LN	I+LN	2	2
D05	1S	LN	I	LN+2	0
D06	1S	LN	I+LN	2	2
D07	1S	LN	I+LN	2	2
D08	1S	LN	I+LN	2	2
D09		LN	I+LN	2	2
F01	1S	LN	I+LN	2	2
F02	1S	LN	I+LN	2	2
F03	1S	LN	I+LN	2	2
F04	1S	LN	I+LN	2	2
F05	1S	LN	LN+I	2	2
F06	1B	LN	LN+I	2	2
F07	Redouble	LN	LN	2	1
F08	1B	1	I	NOM	0
F09	1S	LN	I	NOM$	0
F10	1S	LN	I+LN	2	2
F11	1S	LN	NOM$+LN	2	1
F12	Redouble	mention: impossible			0
F13	1S	LN	LN	2	1
F14		LN	I+LN	2	2
F15	1S	LN	LN	2	1
F16	1B	LN	I	NOM$	0
F17	1S	LN	I+LN	2	2

2.2. Réponses à la question A de l'exercice 2

I.R. (indice de réussite vaut **0** si la structure de donnée n'est pas comprise, **1** si la structure est comprise, mais qu'il y a des erreurs, en particulier dans le calcul d'un argument numérique ou dans l'emploi de la concaténation, **2** si la réponse est correcte).

Num	Posi A	Longueur A	Position B	Long. B	Position C	Long. C	Calcul R	I.R.
B01	1	N-1	N	N-1	N+N	L	A+B+(NxC)	0
B02	R	A	R	B	R	C	A+C+B	0
B03	LEN.(X-1	N	L-L(A)-1	1	Le.(X)-1	1	A,B,C mais en désordre	0
B04	1	L/2	L/2	L-1	L	1	A+B+ INSERT(C,R,N)	1
B05	1	N-1	N	L-(N+1)	N	1	A+C+B	1
B06	1	N-1	N	L-(N+1)	N	1	A+C+B	1
B07	N	R	N	R	N	R	A,B,C	0
B08	L	N	L	N	L	N	A,B,C	0
B09	1	N-1	N	L-(N+1)	L	1	Concat(A,C,B)	1
B10	1	L:2	L:2	L:2	L-1	1	insert('C',X,N)	0
B11	1	3	4	3	7	1	COPY(X,N,1)	0
B12	WRITELN 1ère partie		WRITELN 2nde partie		WRITELN dernier caractère			0
B13	L	N	L	N	L	N	COPY(A,B,C)	0
B14	L	N	L	N	L	N	COPY(A;B;C)	0

B15	1	L/2	L/2	L-1	L	1	A+B+ INSERT(C,R,N)	0
B16	L	N	A	L-1	B	L	A+C	0
B17	1	N	N+1	L-1	L-1	1	C,A,B	0
B18	1	3	4	3	5	1	A,B,C	0
B19	1	3	4	3	7	1	BONJOUR	0
B20	L-1	N	L-L(A)-1	1	L-1	1	COPY(A+B ,C+1,N)	0
B21	L	N	L+N	N	L-1	1	insert(X,C,N)	0
B22	L	N	L+N	N	L-1	1		0
B23	L	N	L+N	N	L-1	1	(X,C,N)	0
B24	L	N	L+N	N	L-1	1	X,C,N	0
B25								0
B26	1	3	1	3	1	1	R	0
D01	1	N-1	N	L-N	L-1	1	A+C+B	1
D02	L DIV 2		L DIV 2	L-1	L-1	1	A+B+C	0
D03		LENGTH(x)				1	A+C+B	1
D04	1	N-1	A+1	L-(A+1)-1	L-1	1		1
D05	L	1						0
D06	1	N-1	N	L-1	L	1	A+C+B	1
D07	1	L DIV N	A+1	L DIV	B+1	L MOD N	A+B+C	0
D08	0	N	N	L-1	L-1	1	A+C+B	1
D09	L DIV2	1	L DIV 2	L-1	L-1	1	RANDOM L-1	0
F01	1	N-1	N	L-N	L-1	1	A$;C$;B$	1
F02	1	N-1	N	L-N	L	1	A$+B$+C$	1
F03	1	N-1	L-1	1	L	1	A$+C$+B$	1
F04	1	N-1	N	L-N	L	1	A$+C$+B$	2
F05	1	N-1	N	L-1	L	1	A$+C$+B$	1
F06	1	N-1	N	L-N	L	1	A$+C$+B$	2
F07	Left(x$)	N-1	L/2	N	right(X$)	1	A$+C$+B$	1
F08	N	1	N	A$	N	B$	A$+B$+C$	0
F09	1	N	A$	N	B$	N	C$+A$+B$	0
F10	1	N-1	N	L-1-N	L	1	A$+C$+B$	1
F11	N-1	L	N+1	L	N	1	A$+C$+B$	0
F12	1	N-1	1	2N-2	L-1	1	A$+C$+B$	1
F13	1	N	N	L-N	L-1	1	A$+C$+B$	1
F14	1	N-1	1	L-N-1	1	L-N	A$+B$+C$	0
F15	1	L-N	N	L-N	L-1	1	A$+B$+C$	1
F16	L-3	L	A$-3	L	L-(L-1)	L	C$+A$+B$	0
F17	1	N-1	N	L-1	L	1	A$+C$+B$	1

2.3. Réponses à la question B de l'exercice 2

- en avant dernière colonne C.: classement utilisé dans le dépouillement :

  A: pas de réponse

  C: réponse correcte

  F: algorithme d'affichage des caractères dans l'ordre inverse

  I: algorithme de calcul correct mais sans rapport avec l'énoncé

M: retournement par moitié

N: séparation en 3 sous-chaîne, mais pas d'itération

P: problème compris, mais erreur dans l'itération

Q: algorithme sans rapport avec l'énoncé (erroné)

- en dernière colonne,I. (indice de réussite) vaut **0** si le procédé indiqué dans l'énoncé n'est pas compris, **1** si le procédé est compris, et s'il y a des erreurs, en particulier l'oubli de l'affectation du résultat à la donnée à chaque tour de boucle, **2** si la réponse est correcte.

Num	programme et éventuellement commentaire	C	I
B01	REPEAT L=L-1 A:=COPY(X,1,L-1)B:=COPY(X,L-1,1) R:=B+A WRITELN(R) UNTIL	P	0
B02	A:=copy(X,R,A) B:=copy(X,R,B) C:=copy(X,R,C) R:=C+B+A	N	0
B03	inverse[0]:=chaine[0] for N=L down to 1 do inverse[L+1-N]:=chaine[N]	I	0
B04	For I:= (L/2) down to 1 do a:=copy(X,I,1)+A FOR I:=(L-1) down to L/2 do B:=copy(X,I,1)+B C:COPY(X, L,1) R:=C+B+A	M	0
B05		A	0
B06		A	0
B07	N:=N:1 A:=COPY(X,N-1,R) B:=COPY(X,L-1,R) C:=COPY(X,L-1,R) R:=C;B;A	N	0
B08	G:COPY(N:1) F:COPY(N:2) E:COPY(N:3) C:copy(N:4) B:copy(N:5) A:COPY(N=6)	N	0
B09	ch2:='' for i:=1 to L do ch2:=concat(copy(ch1,I,1),ch2)	I	0
B10	L:=lenght(x) repeat L:=L-1 A:=copy(X,L,1) writeln(A) until L:=0	F	0
B11	POS:=N-1 I:=0 REPEAT I:=I+1 UNTIL R=COPY(X,POS,1)	Q	0
B12		A	0
B13	for I:=1 to length(R) do R:=copy(X,I,1)+ R	I	0
B14	for i:=1 to length(chaine1) do chaine2:=copy(chaine1,I,1)	Q	0
B15	for I=1 to L/2 do A:=copy(X,I,1)+A for I:=L-1 down to L/2 doB:=copy(X,L/2,L-1)+B C:=copy(X,L,1) R:=A +B+C	M	0
B16	A:=COPY(X,L,L-1) B:=COPY(X,A,C) C:=COPY(X,B,N-1) R:=A+B+C je peux utiliser le programme ci-dessus en additionnant C+B+A (à l'envers). On fait tout à l'envers.	N	0
B17	pos:=position(X;chaine) A:=copy(X;1;1) B:=copy(X;2;1) C:=copy(X;3;1) repeat Z:=copy(X;pos+1;1) until pos=1 R:=Z,C,B,A	P	0
B18	R:=0 repeat R:=copy(X,length(X)-1,1)+R length(X):=length(X)-1 until length(X):=0	Q	0
B19		A	0
B20		A	0
B21	for I:=1 to L do R:=copy(X,I,1)+R	I	0
B22	REPEAT R:=COPY(X,LENGTH(X)-1) X:=DELETE(X,LENGTH(X)-1,1) UNTIL R=' '	Q	0
B23	Readln(N) A:=copy(X-1,1) B:=copy(X+N,N) C:=copy(X,L,N) R:=X,C,N	N	0
B24	readln(N) A:=copy(X-1,1) B:=copy(X+N,N) C:=copy(X,L,N) R:=	N	0
B25	REPEAT A:=COPY(LENGTH(X)) B:=A DELETE(A) A:=COPY(LENGTH(X)) WRITELN(B+A) UNTIL X=0	Q	0

B26	for I:=1 to length(chaine) do chaine2:=copy(chaine1,I,1)+chaine2	I	0
D01	for I=1 to length(X) do A[I]:=copy(X,I,1) for B:=I down to 1 do R:=R+A[B]	I	0
D02	chaine inv:='' for I=1 to length(chaine) do chaine inv:=copy(chaine,I,1)+chaine inv	I	0
D03		A	0
D04		A	0
D05		A	0
D06	R:='' for I:=1 to N begin A:=copy(X,N-I,1) R:=R+A end	I	0
D07	A:=COPY(X,1,L DOWN TO 1)	N	0
D08	WHILE I<=L DO BEGIN I:=COPY(X,L-I,1) WRITE(I) I:=I+1	F	0
D09	FOR I ← 1 LENGTH(X) DO BEGIN XINV ←COPY(X,I,1) + XINV END	C	2
F01	L=LEN(X$) FOR N=1 TO L-1 A$=MID$(X$,1,N-1) B$=MID$(X$,N,L-N) C$=MID$(X$,L-1,1) X$=A$;C$;B$ NEXT N	I	0
F02	N=1 L=LEN(X$) FOR I=1 TO L A$=MID$(X$,1,N_1) B$=MID$(X$,N,L-N) C$=MID$(X$,L,1) R$=A$+B$+C$ NEXT I donner à N la valeur 1 et en rajoutant une boucle permettant de ramener plusieurs fois la dernière..	P	1
F03	FOR N=1 TO LEN(X$)-1 A$=MID$(R$,1,N-1) B$=MID$(R$,N,L-1) C$=MID$(R$,L,1) R$=A$+B$+C$ (X$=R$rayé) NEX Il suffit de lui rentrer pour valeurs de n successivement 1 2 3 (...)de redonner à X$ le résultat à chaque opération.	P	1
F04	N=1 REPEAT A$=mid$(X$,1,N-1) B$=mid$(X$,N,L-N) C$=mid$(X$,L,1) Y$=A$+C$+B$ N=N+1 UNTIL N=L Il faut que le dernier caractère soit mis à la place correspondant N en incrémentant N à chaque passage.	P	1
F05	FOR I=1 TO L A$=MID$(X$,L,1) R$=R$+A$ X$=R$ NEXT I On prend la dernière lettre et on la met au début, cela pour tout le mot.	Q	0
F06	FOR I=1 TO LEN(X$) A$=MID$(X$,+I,1) R$=A$	Q	0
F07	FOR I=1 TO LEN(X$) B$=B$+MID$(X$,LEN(X$)-I+1,1) NEXT	I	0
F08		A	0
F09	FOR I=1 TO LEN(A$) renverser la chaîne avec mid$(Y$ Next	A	0
F10	for Y=len(X$) to 1 step -1 a$=mid$(X$,Y,1) b$=b$+A$ next	I	0
F11	FOR I=1 TO L A$=MID$(X$,L,1) NEXT	Q	0
F12		A	0
F13		A	0
F14		A	0
F15	FOR I=1 TO LEN(X$) I=I+1 R$= MID$(X$,I,1)	Q	0
F16		A	0
F17	for I=1 to L A$=mid$(X$,1,N-1) B$=mid$(X$,I,L-1) C$=mid$(X$,L,1) X$=A$+C$+B$ next on obtiendra un retournement si on met X$=A$+C$+B$ au lieu de R$=...qui est une autre chaîne	C	2

Annexes au chapitre 9

Première série: compréhension des arguments de la fonction sous-chaîne, problème simple de "recherche translation"

ALPHA1: à partir de la chaîne ABCDEFGHIJKLMNOPQRSTUVWXYZ, calculer une lettre de l'alphabet connaissant son rang

L'énoncé est donné sous la forme d'un programme à compléter:

```
REM programme ALPHA1 on veut un programme tel que:
REM
REM L'utilisateur entre un nombre compris entre 1 et 26;
REM l'ordinateur affiche la lettre de l'alphabet de rang correspondant
REM
REM exemple: entrée: 10  réponse J
REM    entrée:  1  réponse A
REM
REM complète le programme:
REM
LET ALPHABET$="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
INPUT N: REM entrée d'un nombre au clavier
LET LETTRE$ = MID$(.........
PRINT LETTRE$
END
```

ALPHA2: à partir de la chaîne ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz, distinguer Majuscules et Minuscules

L'énoncé est donné sous la forme d'un programme à compléter:

```
REM programme ALPHA2  on veut un programme tel que:
REM
REM L'utilisateur entre un nombre compris entre 1 et 26;
REM
REM l'ordinateur pose la question  "MAJUSCULES ?"
REM si l'utilisateur répond OUI, l'ordinateur affiche la lettre de
REM l'alphabet de rang correspondant, en MAJUSCULES
REM
REM si la réponse n'est pas OUI, l'ordinateur affiche la minuscule
REM
REM complète le programme:
REM
LET ALPHABET$="ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz"
INPUT N: REM entrée d'un nombre au clavier
INPUT REP$
IF REP$="OUI" THEN                    ELSE
PRINT LETTRE$
END
```

ALPHA3: recherche inverse (calcul itératif du rang d'un caractère donné dans la chaîne ABCDEFGHIJKLMNOPQRSTUVWXYZ)

L'énoncé est donné sous la forme d'un programme à compléter:

```
REM programme ALPHA3  on veut un programme tel que:
REM
REM  L'utilisateur entre une lettre  de l'alphabet (MAJUSCULE).
REM  l'ordinateur affiche la position de la lettre dans l'alphabet
REM
REM exemple: entrée: J  réponse: 10
REM
LET ALPHABET$="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
INPUT LETTRE$: REM entrée d'une lettre au clavier
REM Complète le programme
```

ALPHA4: Recherche-translation dans la chaîne ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz: une majuscule étant donnée, calculer la minuscule

L'énoncé est donné sous la forme d'un programme à compléter:

```
REM programme ALPHA4  on veut un programme tel que:
REM
REM  L'utilisateur entre une lettre  de l'alphabet (MAJUSCULE).
REM  l'ordinateur affiche la minuscule correspondante.
REM
REM exemple: entrée: J  réponse: j
REM
LET ALPHABET$="ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz"
INPUT LETTRE$: REM entrée d'une lettre au clavier
REM Complète le programme
```

2ème série: Recherche-Translation avec arguments plus généraux

PIANO1: à partir de la chaîne "DO C RE D MI E FA F SOLE LA A SI B", recherche de la notation anglaise d'une note de la gamme

L'énoncé est donné sous la forme d'un programme à compléter:

```
REM programme PIANO1
REM Les notes de la gamme sont codées en BASIC par des lettres
REM Ainsi l'instruction PLAY "CCCDECEDDC" jouera la mélodie
REM      DO DO DO RE MI RE DO MI RE RE DO

REM L'exercice consiste à construire un programme tel que:
REM L'utilisateur entre une note de la gamme (de DO à SI)
REM l'ordinateur joue la note .

REM on utilise la chaîne gamme$ :

gamme$="DO C RE D MI E FA F SOLE LA A SI B "

REM complète le programme; tu peux faire une boucle pour calculer le
rang
REM de la première lettre de note$ dans gamme$
```

```
print "NOTE    ";
INPUT note$
l=len(note$)
 ....................

lettre$= mid$(gamme$,........

play lettre$
end
```

PIANO2: codage d'une mélodie à partir de l'entrée de notes par l'utilisateur

L'énoncé est donné sous la forme d'un programme à compléter:

```
REM programme PIANO2
REM on veut maintenant que l'utilisateur puisse entrer une série de
REM notes de façon que l'ordinateur joue la mélodie correspondante.

REM quand l'utilisateur a entré sa dernière note, il entre FIN

REM le programme ci-dessous est constitué d'une boucle d'entrée des
REM notes par l'utilisateur.
REM insère la boucle que tu as construite pour le programme précédent.
REM de façon à construire la chaîne melodie$

gamme$="DO A RE B MI C FA D SOLE LA F SI G FIN "
melodie$=""
DO
 print "NOTE ?    ";
 INPUT note$
 l=len(note$)
 ..............

 ..............
loop until note$="FIN"
play melodie$
```

3ème série Calcul sur des ordinaux, fonction longueur, conditionnelles à plusieurs variables

TESTC1: comparaison première et dernière lettre d'un mot

L'énoncé est donné sous la forme d'un programme à compléter:

```
REM programme TESTC1; on veut un programme tel que:
REM l'utilisateur entre un mot au clavier; si la
REM première et la dernière lettre du mot sont les mêmes,
REM l'ordinateur affiche "OUI", sinon il affiche "NON".
REM
REM Complète le programme.
REM
INPUT MOT$: REM entrée d'un mot au clavier
LET L = LEN(MOT$): REM L est la longueur du mot
LET A$ = MID$(MOT$,...,...):REM A$ est la première lettre du mot
LET B$ = MID$(MOT$,...,...):REM B$ est la dernière lettre du mot
```

```
.........................
END
```

TESTC2: comparaison première, dernière et lettre du milieu

L'énoncé est donné sous la forme d'un programme à compléter:

```
REM programme TESTC2; inspire-toi du programme TESTC1
REM pour écrire un programme qui résout le problème suivant:
REM
REM L'utilisateur entre un mot au clavier, de LONGUEUR IMPAIRE
REM Si la PREMIERE lettre, la DERNIERE, et celle du MILIEU sont
REM égales, alors l'ordinateur affiche "OUI", sinon il affiche "NON".
REM
REM exemple, entrée: ECOLE   réponse: NON
REM  entrée: AVALE   réponse: NON
REM  entrée: ELEVE   réponse: OUI
REM  entrée: SEVERES  réponse: NON
REM
REM Complète le programme
REM
INPUT MOT$: REM entrée d'un mot au clavier
LET L = LEN(MOT$): REM L est la longueur du mot
LET A$ = MID$(MOT$,...,...):REM A$ est la première lettre du mot
LET B$ = MID$(MOT$,...,...):REM B$ est la dernière lettre du mot
```

TESTC3: comparaison (itérative) des caractères symétriques d'un mot

L'énoncé est donné sous la forme d'un programme à compléter:

```
REM programme TESTC3;   on veut un programme tel que:
REM l'utilisateur entre une phrase au clavier;
REM
REM l'ordinateur compte le nombre  de paires de caractères PLACES
REM SYMETRIQUEMENT (exemple: le premier et le dernier; le second
REM et l'avant dernier...), mais DIFFERENTS
REM
REM exemples: ELU PAR CETTE CRAPULE  réponse: 6
REM
REM    ELUPARCETTECRAPULE  réponse: 0
REM
INPUT CHAINE$: REM entrée d'une chaîne au clavier
LET L = LEN(CHAINE$): REM L est la longueur de la chaîne
```

TESTC4: comparaison (itérative) des caractères symétriques d'une phrase (en sautant les espaces)

L'énoncé est donné sous la forme d'un programme à compléter:

```
REM programme TESTC4;   on veut un programme tel que:
REM l'utilisateur entre une phrase au clavier;
REM
REM l'énoncé est le même que dans l'exercice précédent, mais ici on
REM ne considère pas les espaces entre mots.
REM
REM on considérera qu'il n'y a pas plus d'un espace entre deux mots
REM
REM exemples: ELU PAR CETTE CRAPULE  réponse: 0
REM
```

```
REM     A LAVAL A NOEL LEONA L AVALA: réponse 0
REM
INPUT PHRASE$: REM entrée d'une phrase au clavier
I=1: J= LEN(PHRASE$):REM valeur des positions de la première paire
```

4ème série: Typage des données, fonctions de conversion de type

INSEE1: contrôle de cohérence du numéro INSEE. Calcul du département de naissance

Enoncé:

Le numéro INSEE est attribué à toute personne résidant en France; il est constitué de 13 chiffres.
exemple: 1480575114227

- le premier chiffre est 1 pour un homme, 2 pour une femme.
- les deux chiffres suivants sont les deux derniers chiffres de l'année de naissance (ici année de naissance 1948).
- les deux chiffres suivants désignent le mois de naissance (01 pour janvier, etc...).
- les deux chiffres suivants désignent le département de naissance (ici 75: PARIS).
- les deux groupes de trois chiffres suivants représentent la commune de naissance et le numéro d'inscription sur le registre d'état-civil

Le programme INSEE.BAS, sur la disquette permet d'affecter à la variable INSEE$ un numéro à 13 chiffres, tiré au hasard parmi les 6 numéros qui forment la chaîne LISTE$.
On te demande de compléter le programme INSEE.BAS:

- le programme fait les contrôles suivants:
  - le premier chiffre est 1 ou 2.
  - les 4ème et 5ème chiffres forment bien un mois de l'année (de 01 à 12).
- le programme affiche le numéro du département de naissance

INSEE 2: calcul de l'age, un numéro étant donné (utilisation d'une fonction de conversion de type)

Enoncé:

il existe une fonction VAL qui fait passer d'une chaîne constituée de chiffres à un nombre . Ce nombre est celui dont l'écriture en base 10 est constituée des chiffres de la chaîne. Ce nombre est alors utilisable dans des calculs tels que sommes, produits...
1) Charge et exécute le programme qui est sur la disquette sous le nom TESTVAL (pour voir si tu as compris ce qui précède);
2) En reprenant les indications ci-dessus, complète le programme INSEE.BAS de façon que le programme affiche l'age du titulaire du numéro au premier janvier 1989.
Attention: si le nombre constitué des 2 derniers chiffres de l'année de naissance est supérieur à 89, cela signifie que l'utilisateur est né avant 1900.

INSEE 3: calcul du numéro INSEE à partir de données entrées par l'utilisateur (concaténation; variables et constantes)

Enoncé

Ecris un programme tel que:
- l'utilisateur entre le sexe: (Masculin ou Féminin)
- l'utilisateur entre le mois et l'année de naissance
- l'utilisateur entre le département de naissance.
- l'ordinateur calcule et affiche le numéro INSEE;

REMARQUE: les 6 derniers chiffres n'étant pas connus seront remplacés par la chaîne "xxxxxx".

INSEE4: calcul de la clé de contrôle (coordination entre la représentation du numéro INSEE comme suite de chiffres, et sa valeur numérique)

Enoncé:

Le numéro INSEE est accompagné d'une clé de contrôle, constituée de 2 chiffres, de 01 à 97, permettant de déceler une erreur éventuelle (57 pour l'exemple ci-dessus).
C'est la différence de 97 et du reste dans la division du nombre formé par les 13 chiffres par 97. On souhaite ici faire un programme permettant, à partir d'un numéro INSEE d'obtenir sa clé de contrôle.
Le Basic permet de disposer d'une fonction MOD (modulo); A et B étant des nombres, A MOD B est le reste dans la division de A par B;
Mais les nombres ne peuvent être supérieurs à 32 767. C'est pourquoi on divisera le numéro INSEE en trois tranches:
- la première, constituée des cinq premiers caractères, donnera le reste R1,
- la seconde, constituée des quatre caractères suivants, donnera le reste R2
- la dernière, constituée des quatres derniers caractères, donnera le reste R3,

Ayant 10 000 MOD 97 = 9, le reste du nombre à treize chiffres est le même que celui de R1 * 81 + R2 * 9 + R3 . Il reste à le calculer, ainsi que son complément à 97.

5ème série: Utilisation de la concaténation. Coordination de structures algorithmiques complexes

L'énoncé est donné sous la forme d'un programme à compléter:

```
REM Un médecin propose des consultations sans rendez-vous.
REM Les patients arrivants sont dirigés vers la salle d'attente;
REM lorsqu'une place est libre pour la consultation, le médecin appelle
REM le patient qui attend depuis le plus longtemps;

REM Pour se faciliter la tâche, le médecin voudrait un programme sur son
REM ordinateur qui fonctionne de la façon suivante:

REM Lorsqu'un patient arrive, il tape le signe + (Return)
REM puis une lettre de l'alphabet qui lui est propre
REM (on suppose qu'il n'y a pas plus de 26 patients, pour simplifier).

REM Les lettres correspondant aux patients présents dans la salle d'attente
REM sont mémorisées dans la chaîne de caractères liste$, dans l'ordre d'arrivée.

REM Lorsque le médecin a une place libre pour la consultation,
REM il tape le signe - (Return) , l'ordinateur affiche la lettre
```

```
REM correspondant au patient qui attend depuis le plus longtemps, et
REM met la liste à jour.

REM Lorsque la consultation est terminée, le médecin tape le
REM   signe / (Return) pour arrêter le programme.

REM Complète le programme.(tu peux remplacer les ..... par plusieurs lignes.


LET liste$="":REM la liste des patients dans l'ordre d'arrivée.
L=0   : REM L est le nombre de patients dans la salle d'attente.

do
      do
            Print "Arrivée d'un patient (+), place libre(-), ou fin (/)?"
               INPUT rep$
      loop until rep$="+" or rep$="-"   or rep$="/"
      cls

      if rep$="-" then
      ..................................
      endif

      if rep$="+" then
      ..................................
      endif
loop until rep$="/"

print "consultation terminée"
print "restent dans la salle d'attente les patients ";liste$

end
```

Annexes au chapitre 10

Compte-rendus des 9 séances de résolution analysées au chapitre 10. (Rappel: les énoncés complets sont en annexe 9)

Séance 1: 8 mars; durée 1h.: présentation des notions par le professeur, et résolution de ALPHA1 (à partir de la chaîne ABCDEFGHIJKLMNOPQRSTUVWXYZ, calculer une lettre de l'alphabet connaissant son rang)

La première demi-heure consiste en des rappels sur les chaînes de caractères et une présentation par le professeur de la fonction "sous-chaîne" . Dans la partie "rappels", le professeur insiste sur:

- la différence entre les variables numériques (sur lesquelles on peut faire des calculs tels que addition et multiplication) et les variables chaînes (que l'on veut pouvoir utiliser pour entrer du texte, c'est-à-dire un morceau de phrase, ou un mot); la différence est faite entre la variable numérique A valant 127, et la variable chaîne valant "127", du point de vue des opérations valides sur chacune de ces variables,
- la notion de caractère: en particulier l'espace est un caractère, les chaînes "ABC" et "ABC " sont différentes,
- la possibilité de réaliser des tests d'égalité sur les chaînes, possibilité qui a été exploitée pour contrôler des réponses entrées par l'utilisateur dans un programme.

Concernant la fonction MID$ (sous-chaîne), les points suivants sont soulignés:

- la nécessité de "faire des choses" sur les chaînes de caractères, par analogie avec les opérations qu'ont sait faire sur les nombres, et en relation avec des problèmes qu'on voudrait résoudre (crypter un message selon une loi...).
- le caractère fonctionnel de "sous-chaîne", avec les deux aspects suivants:
    - une analogie avec les fonctions rencontrées en mathématiques (la fonction qui à x associe x puissance 2), qui permet de préciser les notions de "variable", (c'est à dire l'information en entrée) et de résultat, ainsi que le "procédé de calcul" que résume la fonction; le professeur souligne que la fonction MID$ est à trois variables, contrairement aux fonctions rencontrées par les élèves en mathématiques.
    - la nécessité d'"utiliser le résultat rendu par MID$" soit dans une affectation, soit pour un affichage à l'écran.

Le travail des élèves: pendant la 1/2 heure restante, les élèves résolvent les problèmes **ALPHA1** et **ALPHA2**. Natacha est absente pendant les premières séances; **Karine** travaille donc seule.

ALPHA1: il s'agit de calculer la ième lettre de l'alphabet, la suite ordonnée des lettres de l'alphabet étant affectée à une variable chaîne ALPHABET$. Les 3 élèves observés (**Karine, Jérôme , Arnaud**) ne rencontrent pas de difficulté; ce problème est résolu sans aide.

ALPHA2: ce problème nécessite l'écriture de la fonction sous-chaîne avec un second argument variable suivant que l'utilisateur désire une majuscule ou une minuscule . La variable chaîne ALPHABET$ contient ici la suite ordonnée des lettres majuscules de l'alphabet suivie de la suite ordonnée des lettres minuscules de l'alphabet . Dans un cas, le second argument de la fonction MID$ que les élèves doivent écrire, résulte d'un calcul: on voit apparaître des écritures montrant des difficultés d'acquisition concernant la fonction sous-chaîne:

- **Karine**: IF REP$="O" THEN LETTRE$="MAJUSCULES" puis, après un temps d'hésitation ("je crois que je vais faire une bêtise") MID$(ALPHABET$,N,MAJUSCULES) ELSE LETTRE$=MID$(ALPHABET$,N,MINUSCULE), puis, suite à un dialogue avec le professeur: MID$(ALPHABET$,N,1) ELSE LETTRE$=MID$(ALPHABET$,N,N+26).
- **Jérôme** et **Arnaud**: LETTRE$ se voit de même affecter MID$(ALPHABET$,N,N+26) dans la partie suivant "ELSE" de l'alternative;

Séance 2: 13 mars; durée: 1h30 Résolution de ALPHA3 puis de ALPHA4

ALPHA3: il s'agit de trouver le rang d'une lettre dans l'alphabet, en parcourant une chaîne ALPHABET$ ayant comme valeur "ABCDEFGHIJKLMNOPQRSTUVWXYZ".
ALPHA4: ALPHABET$ a cette fois comme valeur une chaîne formée des 26 lettres de l'alphabet majuscules, suivies des 26 lettres de l'alphabet minuscules; on demande d'entrer une majuscule, et d'afficher la minuscule correspondante. **Karine** (travaille seule en l'absence de **Natacha**). Je donne les indications suivantes:

- utiliser une boucle DO...LOOP
- écrire le test d'arrêt
- pour ce test, utiliser la fonction MID$

Le test d'arrêt est commencé sous forme LETTRE$=MID$(ALPHABET$, L'élève hésitant sur le second argument, j'indique que, de façon générale, quand une quantité n'est pas connue, on peut la désigner par une variable; l'écriture devient: LETTRE$=MID$(ALPHABET$,X,Y) puis est corrigée spontanément en LETTRE$=MID$(ALPHABET$,X,1).

Suite à une discussion sur l'action à répéter dans le corps de boucle, l'élève écrit LET X+1, ce qui a pour elle le sens de "incrémenter X". J'explique la syntaxe de LET (pas de LET sans "="), et sa signification. L'élève propose LET ALPHABET$=X+1 puis LET X=X+1. L'exécution réussit, ce qui semble étonner l'élève.

Résolution de **ALPHA4**.

La première écriture produite spontanément est: PRINT MID$(ALPHABET$, + 26,1); l'élève déclare qu'elle ne sait pas quoi mettre avant le + 26 dans le second argument; J'indique, comme dans le problème précédent de mettre une variable X, et d'écrire le calcul de X au dessus. L'élève propose comme calcul: LET LETTRE$=X et explique: il faut lui dire que c'est la Xième lettre. J'attire plusieurs fois l'attention sur le fait que le calcul de X a été résolu au problème précédent.

L'élève écrit sur le modèle du problème précédent: LET X=X+1 LOOP UNTIL X= MID$(ALPHABET$,X+26,1) L'élève obtient l'erreur de compilation: "*opérateur*

de comparaison attendu dans une expression booléenne"; en effet, les booléens étant un sous-type du type numérique en **BASIC**, la comparaison à la variable numérique X du résultat d'une expression booléenne aurait un sens; le compilateur considère qu'une telle expression est commencée par l'écriture d'une expression à valeur chaîne:`MID$(ALPHABET$,X+26,1)` et attend l'opérateur de comparaison qui doit suivre pour former une expression booléenne. A la suite de cette erreur la deuxième ligne est transformée en: `LOOP UNTIL LETTRE$= MID$(ALPHABET$,X+26,1)` ce qui conduit à une erreur d'exécution: "*Appel de fonction invalide*"; la condition n'étant jamais remplie, `X + 26` devient supérieur au nombre de caractères de `ALPHABET$`, entraînant un appel illégal à la fonction `MID$`.

J'attire à plusieurs reprises l'attention sur la nécessité de distinguer les deux parties: calcul de X, et calcul du résultat, et sur le fait que chacun de ces problèmes a été résolu précédemment, mais la séance se termine sans progrès notable dans la résolution.

Jérôme Arnaud:

Résolution de **ALPHA3**: Les élèves commencent la boucle sous la forme:

```
DO LET LETTRE$=MID$(ALPHABET$,X,1)
```

Il est possible que cette expression résulte d'une confusion entre le corps de boucle et la condition d'arrêt. Les élèves ne progressant pas davantage, je leur donne deux indications:

- il n'est pas opportun de changer la valeur de `LETTRE$`, car la lettre dont le rang est à rechercher sera alors perdue,
- écrire la condition d'arrêt.

Suite à la première indication, `LETTRE$` est changé en `LETTRE1$`; suite à la seconde, la condition d'arrêt est écrite: `LETTRE1$=LETTRE$` . J'attire l'attention sur la nécessité de faire évoluer X. Les élèves écrivent l'instruction sous la forme `LET X + 1`; j'attire leur attention sur la signification de l'affectation (qui suppose une variable à laquelle une valeur est affectée); l'instruction est corrigée en `LET NOMBRE$=X+1` . Il semble que cette écriture soit la traduction mot à mot d'une phrase que j'ai prononcé:" X+1 doit être affecté à un nombre". J'indique que c'est X qui doit changer, et les élèves écrivent finalement `LET X=X+1` .

Mais comme cette instruction est placée dans le corps de boucle APRES le calcul de `LETTRE1$`, le premier passage dans la boucle conduit à une erreur "*Appel de fonction invalide*"(X n'ayant pas reçu de valeur initiale, **BASIC** lui affecte zéro, et zéro comme second argument de `MID$` est une erreur).Les élèves écrivent avant la boucle l'instruction d'initialisation. Le programme a alors la forme suivante:

```
LET X=1
DO
 LET LETTRE1$=MID$(ALPHABET$,X,1)
 LET X=X+1
LOOP UNTIL LETTRE1$=LETTRE$
PRINT X
```

Il n'y a plus d'erreur d'exécution, mais le test d'arrêt n'étant pas cohérent avec la séquentialité dans le corps de boucle, le résultat est systématiquement supérieur de 1 au résultat attendu. Les élèves changent alors l'ordre des instructions du corps de boucle. Ils font des essais d'exécution qui leur donnent satisfaction. Sur ma suggestion , il essaient avec la lettre "A" en entrée; le programme conduit alors à une boucle sans

fin; en effet, le changement dans l'ordre des instructions du corps de boucle a été effectué sans que l'initialisation ait été modifiée en conséquence, et donc X n'est jamais égal à 1 quand il est utilisé pour le calcul de A$. Les élèves modifient alors l'initialisation.

Résolution de **ALPHA4** La recherche est abordée pendant le dernier quart d'heure; je donne aux élèves l'indication de se servir du programme précédent (**ALPHA3**) et de le continuer, mais aucun résultat appréciable n'est obtenu à la fin de la séance.

Séance 3: 20 mars; durée 1h15 ; Résolution du problème PIANO1.

Il s'agit, à l'aide de la chaîne GAMME$ ayant pour valeur "DO C RE D MI F FA G SOLH LA A SI B FIN " d'entrer une chaîne de deux ou trois caractères (DO, RE ... SI) et de calculer la lettre correspondante (C ,D , ..., A, B) et de la faire jouer par l'ordinateur.

Groupe **Jérôme Arnaud** Le travail ne démarrant pas , je donne l'indication suivante:

- envisager la façon dont peut procéder la machine quand on entre la note SOL.
- considérer que la seule connaissance dont dispose l'ordinateur est la chaîne GAMME$

Les indications ne donnent pas de résultat, je tente une analogie avec la recherche de la traduction d'un mot dans un dictionnaire français-anglais, sans plus de succès. Il est alors demandé aux élèves de se concentrer sur l'indication donnée en remarque dans le programme: tu peux faire une boucle pour calculer le rang de la première lettre de NOTE$ dans GAMME$. "Supposons qu'on ait trouvé ce rang, appelons le X; comment est-ce qu'on pourrait obtenir la lettre demandée?" Un élève complète alors LETTRE$= MID$(GAMME$,X+3,1) qui est ensuite modifié, suite à des interventions de l'autre membre du groupe en MID$(GAMME$,X,X+3); puis MID$(GAMME$,X+3,X)

J'attire alors l'attention des élèves sur la signification du troisième argument. Et je fais remarquer avec insistance qu'on connaît ici la longueur de la sous-chaîne. Les élèves reviennent enfin à la formulation LETTRE$=MID$(GAMME$,X+3,1) . J'attire ensuite leur attention sur la nécessité que le programme calcule X, et leur demande de construire une boucle. Après un temps de recherche autonome, les élèves ont écrit le début de la boucle:

```
DO LET x=x+1
```

et s'apprêtent à rédiger l'avertisseur de fin du corps de boucle et la condition d'arrêt **après** le calcul de LETTRE$. Je leur fait remarquer que le calcul de LETTRE$ suppose connue la valeur de X, et leur fait déplacer l'avertisseur d'itération. Les élèves rédigent ensuite la condition d'arrêt sous la forme: LETTRE$=NOTE$, ce qui montre que pour eux, le calcul de LETTRE$ à l'intérieur de l'itération pouvait avoir un sens, l'égalité signifiant pour les élèves, que les écritures représentent la même note , et non que les chaînes ont la même valeur. Le programme s'écrit alors:

```
DO LET x=x+1
UNTIL LETTRE$=NOTE$
LETTRE$=MID$(GAMME$,X+3,1)
```

Un essai d'exécution conduit à une itération sans fin (LETTRE$ n'ayant pas reçu d'affectation, le **BASIC** considère cette variable comme ayant la valeur chaîne vide, la condition d'arrêt n'est jamais satisfaite). Il faut alors relancer l'ordinateur, recharger le **BASIC** et le programme; je propose ensuite de concentrer la recherche sur la condition d'arrêt; devant l'absence de réaction des élèves, j'indique de compléter la condition d'arrêt sous la forme: NOTE$=MID$(.....) Après un temps de recherche autonome, les élèves ont écrit la condition d'arrêt sous la forme NOTE$=MID$(GAMME$,X+3,1).

Je corrige le second argument (X au lieu de X+3), puis demande qu'on précise le troisième, en indiquant que la longueur de NOTE$ peut être 2 ou 3. Le fait que cette longueur est variable met les élèves dans l'embarras. **Michael** finit par proposer i; la variable contenant cette longueur ayant comme identificateur l (minuscule), la confusion s'explique. J'indique donc de mettre "1" comme troisième argument. Le programme est ensuite exécuté pour les différentes notes, et considéré comme terminé une dizaine de minutes avant la fin de la séance.

L'opération de concaténation est présentée, et les élèves tentent sans succès pendant le reste de la séance de commencer le problème **PIANO2**.

Karine et **Natacha** (**Natacha** était absente aux séances 1 et 2).

Les élèves posent une question sur l'instruction LET L=LEN...; j'explique qu'il s'agit du nombre de caractères de la chaîne argument. Suit une discussion avec les élèves sur la façon dont l'ordinateur pourrait procéder. J'attire l'attention sur l'indication donnée en remarque, et demande ce qu'est le "rang" de la première lettre, et l'intérêt de le calculer: il y a une certaine confusion chez les élèves: il n'est pas clair que le rang soit un nombre; par contre l'intérêt de le connaître est vu. ("alors c'est X + 3"). Je souligne la séparation du programme en deux parties: le calcul de X, rang de la première lettre de NOTE$ d'une part, et le calcul de LETTRE$ d'autre part, en insistant sur la possibilité de rédiger la seconde partie avant la première. En recherche autonome, les élèves commencent la boucle sous la forme

```
DO
LET X=X+3
UNTIL RANG$=MID$(GAMME$,X,1)
```

Je demande la signification de la variable RANG$; puis n'ayant pas de réponse, les valeurs prises par cette variable; comme **Arnaud-Jérôme** à la séance précédente, les élèves ont cherché une traduction mot-à-mot d'une des indications que j'ai donné en langage naturel. N'ayant toujours pas de réponse, je propose de commencer la condition d'arrêt par NOTE$=MID$(GAMME$... et demande de compléter. En recherche autonome, les élèves modifient la boucle en:

```
DO
LET X= X- 6
until NOTE$=MID$(GAMME$,x,1)
```

Je fais remarquer que le troisième argument ne convient pas et que la longueur peut valoir 2 ou 3. Les élèves proposent d'abord 3, en expliquant que cette valeur conviendra toujours; j'explique que les chaînes "RE" et "RE " sont différentes. Le troisième argument est corrigé. Je fais également corriger le corps de boucle en "LET x=x+5". J'attire ensuite l'attention sur l'initialisation de la variable X: les élèves insèrent l'instruction "LET X=1" avant le début de la boucle. Je fais remarquer que la note "DO" ne sera jamais trouvée, et après quelques hésitations, les élèves corrigent en "LET X=-4"

Séance 4: 22 mars; durée 1h; Résolution de TESTC1

Il s'agit d'écrire un programme testant si la première et la dernière lettre d'une variable chaîne d'identificateur `MOT$` sont les mêmes.

Karine, Natacha

Les élèves complètent les arguments pour la première lettre: `MID$(MOT$,1,1)` puis pour la dernière lettre:`MID$(MOT$, ,1)` **Karine** hésite sur le second argument: `L`, ou `L+1` ? Elle exprime le sentiment "qu'il manque quelque chose"; "qu'il ne va pas comprendre", "que ça nous donne la longueur entière du mot" **Natacha** paraît plus assurée; pour elle `L` représente bien un nombre. Le problème est terminé au bout d' 1/4 h.

Jérôme, Arnaud

L'écriture de la fonction `MID$` pour la première et dernière lettre est obtenue spontanément. Le problème est terminé vers 13h20

Résolution de **TESTC2**: il s'agit ici, dans le cas d'un mot de longueur impaire, de tester si la première lettre, la dernière lettre et celle du milieu sont les mêmes.

Karine Natacha:

Une variable `C$` est introduite, calculée spontanément à l'aide de l'expression: `MID$(MOT$,L/2 + 0.5,1)`. Ayant demandé une explication, j'obtiens la réponse suivante: un caractère ne peut être décimal, se terminer par `0,5`, c'est pour cela qu'on a rajouté 0.5 . Il est probable que les élèves ont fait un essai avec comme second argument `L/2`, et constaté qu'elles n'obtenaient pas le caractère attendu: en effet, au cas où un des arguments n'est pas entier, le **BASIC** le remplace par l'entier immédiatement inférieur (c'est une conversion de type implicite). Elles ont donc introduit `+ 0.5` à titre de correction, en vue d'obtenir un entier.

La partie condition de l'alternative est d'abord écrite `IF A$=B$=C$`. La compilation conduit à l'erreur:"*Opérateur de comparaison attendu dans une expression numérique*"; en effet, le compilateur interprète la partie `A$=B$` comme une expression booléenne, donc numérique (compte-tenu de ce que, dans ce langage, les booléens sont des nombres); de l'autre côté du second signe =, il doit donc trouver une expression numérique; il trouve une variable chaîne; il interprète cette variable, comme le début d'une expression booléenne, donc numérique; ne trouvant pas le reste de l'expression, et en particulier l'opérateur de comparaison qui permettrait de former une expression booléenne, il envoie le message signalé ci-dessus; ce message est évidemment sans rapport avec l'erreur de conception qui a conduit les élèves à cette écriture; il faut remarquer à ce propos que la même expression avec des variables numériques (`IF A=B=C THEN...`) n'aurait pas entraîné d'erreur, ni à la compilation, ni à l'exécution; en effet, `A=B` aurait été compris comme un booléen, donc valant zéro (pour Faux) ou -1 (pour Vrai) et la comparaison entre `A=B` et `C` aurait eu un sens comme comparaison de deux nombres.

Après une explication de ma part sur la nature de l'erreur, l'alternative est d'abord séparée en deux alternatives distinctes:

```
IF A$= B$ THEN PRINT "OUI" ELSE PRINT "NON"
IF B$= C$ THEN PRINT "OUI" ELSE PRINT "NON
```

puis suite à une exécution, cette partie est réécrite sous forme d'une seule alternative avec la condition: `A$=B$ AND B$=C$`. Le problème est terminé en 1/2h.

Jérôme, Arnaud

Une variable C$ est introduite, calculée à l'aide de l'expression: MID$(MOT$,L/2). La partie condition de l'alternative est d'abord écrite IF A$=B$=C$; obtenant la même erreur à la compilation que **Karine-Natacha**, le groupe réécrit la condition sous la forme A$=B$ AND B$=C$. Le programme est alors syntaxiquement correct: l'absence de troisième argument dans la fonction MID$, ainsi que la présence d'un nombre non entier comme second argument ne sont pas considérés comme des erreurs de syntaxe par la version de **BASIC** employée. Mais à l'exécution, sur les exemples de l'énoncé, le programme ne donne pas les réponses attendues.

J'attire l'attention sur l'absence de troisième argument; le programme est corrigé, mais donne les mêmes résultats erronés; n'obtenant pas d'explication, je suggère de mettre une "piste" sur C$; c'est-à-dire de faire afficher C$ en cours d'exécution. On constate ainsi que C$ n'est pas la lettre attendue, mais celle qui précède. Je demande ce que représente L, en tant que variable, puis L/2. Concernant L/2, j'obtiens la réponse: "c'est le milieu du mot". Les élèves admettent que L/2 va être fractionnaire, mais ne comprennent pas pourquoi l'ordinateur choisit la lettre de rang immédiatement inférieur, et non la lettre de rang immédiatement supérieur (ce qui donnerait la bonne réponse). Je dois expliquer ce qu'est une troncature. Le second argument est finalement corrigé en L/2 + 1/2.

Résolution de **TESTC3**: il s'agit de vérifier la qualité de "palindrome" d'une chaîne, en comptant le nombre de paires symétriques différentes. Dans les deux groupes, l'énoncé demande un temps de relecture et d'explication:

Karine, Natacha

La rédaction produite en fin d'heure comporte seulement la recopie des affectations concernant A$, B$ et C$ (respectivement comme première, dernière lettre, et lettre du milieu), ainsi qu'une ligne DO, indiquant que les élèves ont envisagé une itération.

Jérôme, Arnaud

Jérôme et **Arnaud** ne démarrant pas, je leur suggère de faire "varier le caractère" et donc d'employer une variable pour le rang; l'écriture suivante est obtenue:

```
A$=MID$(CHAINE$,X,1)
B$=MID$(CHAINE$,L,1)
```

J'attire l'attention sur le fait que B$ reste égal à la dernière lettre (je n'aperçois pas, à ce moment de l'observation, que l'intention des élèves est de faire varier L). Je guide les élèves pour qu'ils considèrent les caractères intervenant dans les premiers pas de l'itération; ils corrigent la seconde affectation en B$=MID$(CHAINE$,L-X,1) , puis la première en A$=MID$(CHAINE$,X+1,1); de façon spontanée, ils ajoutent les instructions:

```
X=X+1
L=L-1
```

Séance 5: 10 avril; durée 1h30; résolution de TESTC3 (suite)

Karine,Natacha

La première écriture ne comporte pas de boucle , mais les deux lignes:

```
A$= MID$(CHAINE$,L/2+ 0.5,1): REM A$ est le centre de symétrie
X= L/2+ 0.5 + 1
```

Je demande comment l'ordinateur peut "parcourir la chaîne pour faire les comparaisons". **Natacha** indique qu'on compare la première lettre et la dernière, la seconde et l'avant dernière. Je demande d'écrire les affectations permettant d'affecter à A$ la Xième lettre et à B$ la "L-X ième lettre". De façon autonome, le programme suivant est écrit:

```
LET A$= MID$(CHAINE$,X+1,1)
LET B$=MID$(CHAINE$,L-1,1)
IF A$=B$ THEN
```

Les élèves hésitent sur la suite de l'alternative; il semble que **Karine** veuille une action sur X, tandis que **Natacha** voit qu'il faut faire évoluer un compteur; je suggère d'employer une variable "RESULTAT", et **Natacha** écrit RESULTAT= RESULTAT + 1. Les élèves prennent ensuite en compte le fait que l'énoncé demande de compter les paires *différentes*; l'écriture devient:

```
IF A$=B$ THEN          ELSE RESULTAT=RESULTAT + 1
```

Il semble que les élèves aient envie de placer une action comme RESULTAT=RESULTAT+0; je suggère de changer plutôt la partie condition, et la ligne devient:

```
IF A$<>B$ THEN        RESULTAT=RESULTAT + 1
```

Les indicateurs d'itération et la condition d'arrêt sont ensuite écrits de façon spontanée, sous la forme suivante:

```
DO .... LOOP UNTIL L/2
```

L'instruction d'affichage à l'écran du résultat est écrite après la boucle. Le programme est alors syntaxiquement correct, mais faux pour deux raisons:

- les variables X et L n'évoluent pas dans le programme;il est tout à fait probable que dans l'esprit des élèves, l'écriture du second argument dans les expressions:

  ```
  LET A$= MID$(CHAINE$,X+1,1)
  LET B$=MID$(CHAINE$,L-1,1)
  ```

  a pour fonction précisément d'incrémenter X et de décrémenter L par effet de bord, un peu comme le font des instructions d'adressage indexé de certains langages[18] .

- la condition d'arrêt L/2 est une traduction de la formulation naturelle: "on s'arrête à la moitié de la chaîne"; elle se trouve syntaxiquement correcte car , malencontreusement, en **BASIC** les booléens sont compatibles avec les nombres (0 représente FAUX, tout autre nombre représente VRAI); n'étant pas nul, la condition d'arrêt se trouve réalisée dés le premier tour.

- la formulation de la condition d'arrêt ne prend pas en compte le fait que L évolue; dans cette condition, L signifie "la longueur de la chaîne", alors que dans le corps de boucle, L est un index qui évolue.

---

[18] LDA ,X+ en Assembleur 6809, par exemple, charge dans A le contenu de la mémoire d'adresse X, puis incrémente l'index X; c'est ce qu'on appelle la "post-incrémentation"

Les élèves font plusieurs essais d'exécution, en particulier les exemples de l'énoncé; la réponse de l'ordinateur est toujours zéro ou un. Je fais remarquer que rien dans le programme ne fait évoluer les variables X et L, et le corps de boucle est corrigé en:

```
LET A$= MID$(CHAINE$,X,1)
LET B$=MID$(CHAINE$,L,1)
LET X=X+1
LET L=L-1
```

Le corps de boucle est maintenant juste, mais à l'exécution, le calcul de A$ produit l'erreur "*appel de fonction invalide*", à cause de l'absence d'une initialisation de la variable X. (**BASIC** affecte zéro à toute variable numérique non-initialisée, ce qui produit au premier tour de boucle un appel de MID$ avec zéro comme second argument). **Karine** écrit une initialisation sous forme LET X=1; mais **Natacha** insiste pour que **Karine** transforme en LET X=0; une nouvelle exécution produit la même erreur; **Natacha** explique ensuite qu'il faut , pour elle, initialiser X à zéro, car dans le corps de boucle, se trouve l'instruction LET X = X+1

On retrouve le cas classique signalé en particulier par [ROGALSKI 88], dans lequel l'élève ne prend plus en compte la séquentialité des instructions, à partir du moment où ces instructions forment le corps d'une boucle. Après l'erreur d'exécution, **Natacha** accepte l'initialisation LET X=1.

A ce stade de la recherche, il reste l'erreur sur la condition d'arrêt, ce qui conduit aux mêmes résultats erronés à l'exécution: le résultat est systématiquement un ou zéro. J'indique donc de porter l'attention sur cette condition d'arrêt; les élèves la changent en X=L, sans être capable d'en donner une explication; on remarquera que cette condition se vérifie uniquement dans la cas où la longueur de la chaîne est **impaire**; dans le cas où cette longueur est paire, la boucle se termine sur l'erreur "*Appel de fonction invalide*" (quand L devient nul). Il semble qu'après diverses exécutions, les élèves en prennent conscience puisqu'elles modifient la condition en X=L OR X = L + 0.5

Les exécutions donnent la même erreur; les élèves expliquent que dans le cas d'une chaîne de longueur paire, X et L se retrouvent "au milieu, entre deux lettres" , ce qui justifie sans doute le 0.5; je tente d'expliquer que X et L sont des entiers, donc que X = L + 0.5 ne peut être VRAI; mais cette explication ne semble rien évoquer pour les élèves. J'indique alors de "regarder comment évoluent X et L dans le cas de la chaîne "ELU PAR CETTE CRAPULE" "; il est difficile de saisir comment raisonnent ces élèves, car elles s'expriment très peu, mais le mouvement des doigts sur le papier laisse penser qu'elle font évoluer X et L comme **positions** dans la chaîne, sans leur donner de valeur numérique. Leurs hésitations durent une dizaine de minutes; je leur indique à plusieurs reprises qu'elles sont proches du but, car elles semblent parfois penser que l'erreur vient d'une autre partie du programme; la condition est finalement écrite: X=L OR X=L-1, et les exécutions conduisent au succès.

Les élèves passent ensuite le 1/4 d'heure restant à chercher **TESTCH4** (qui consiste à ne pas tenir compte des espaces dans le comptage des paires symétriques), mais sans résultat notable.

Jérôme Michael (**Arnaud** est absent cette semaine; **Jérôme** travaille avec un élève **Michael** qui d'habitude travaille seul).Les élèves travaillent sur le programme écrit par **Michael** à la dernière séance:

```
LET A = L/2
LET X=0
```

```
DO
LET A$= A-X
LET B$= A+X
LOOP UNTIL A$=L AND B$=L
IF A$=B$
```

Ce programme traduit la volonté des élèves de comparer les couples "à partir du milieu", et non "premier-dernier", puis "second-avant-dernier" etc... Le fait de "partir de L/2" restreint l'emploi du programme à une chaîne de longueur paire. Le programme contient des erreurs caractéristiques:

1 - confusion classique entre un caractère et sa position dans la chaîne (qui conduit à une erreur de type à la compilation)

2 - absence d'évolution du paramètre de boucle (ce qui conduit à une boucle sans fin)

3 - erreur sur la condition d'arrêt:à condition de passer sur la confusion signalée plus haut entre caractère et position, B$=L est bien une condition se réalisant en fin de recherche; que penser alors de la condition A$=L ? On attendrait A$=1 , mais pour les élèves A$=L AND B$=L signifie peut-être que "toute la chaîne" a été examinée, selon la confusion signalée dans les études précédentes, concernant la fonction longueur.

4 - début de l'écriture du comptage des couples **à l'extérieur de l'itération**. (difficulté de séquentialité).

La compilation du programme produit une erreur (causée par *l'affectation d'une expression numérique à une variable de type chaîne*); suite à une intervention du professeur, les élèves corrigent, en gardant cependant la même structure de programme (les erreurs 2, 3 et 4 demeurent donc). La compilation se fait alors sans erreur, mais l'exécution conduit à une boucle sans fin. Le professeur intervient à nouveau et fait avec les élèves un tableau de l'évolution des variables:

L	A	B	X	
6			1	
	3	4		A ← A - X
	2	5		B ← B + X
	1	6		fini.

Puis de façon autonome, le programme suivant est écrit

```
LET A = L/2 + 1
LET B = L/2
LET X=1
DO
LET A= A-X
LET B= B+X
LOOP UNTIL A=1 AND B=L
LET A$= MID$(CHAINE$,A,1)
LET B$=MID$(CHAINE$,B,1)
LET N=0
IF A$=B$ THEN N=N+1 ELSE N=N
PRINT N
END
```

La présence d'une affectation sans effet (N=N) comme seconde branche de l'alternative est à rapprocher des hésitations de **Natacha-Karine** pour la même

alternative. Les élèves veulent que le compteur d'occurrence reçoive une valeur dans les deux branches de l'alternative, un peu comme s'il s'agissait d'une forme fonctionnelle de l'alternative, concevable, et même prévue dans certains langages (Multiplan, Dbase, LISP...): affecter à `N SI A$=B$ ALORS N+1 SINON N`. Les élèves n'aperçoivent pas comme évidente l'économie d'une branche de l'alternative permise par la forme impérative de cette alternative.

On remarquera que les erreurs 1, 2, 3 ont été corrigées: ce programme est assez proche syntaxiquement du programme initial, bien que les variables y jouent des rôles différents (`X` est maintenant fixe, alors qu'il semblait destiné à être le paramètre de boucle dans le programme initial). Par contre l'erreur de séquentialité numérotée 4 ci-dessus , demeure: dans leur volonté de séparer l'évolution des pointeurs `A` et `B` d'un côté, et le comptage des couples formés de deux lettres égales, de l'autre, les élèves ont sorti de la boucle ce comptage. Le programme est syntaxiquement correct, mais le résultat est systématiquement zéro ou 1; je suggère alors aux élèves de s'intéresser avec précision à "ce qui doit être dans la boucle". Les élèves passent alors les deux instructions de calcul des caractères, l'initialisation de `N`, et l'instruction alternative en début de boucle, sans donner d'explication:

```
DO
LET A$= MID$(CHAINE$,A,1)
LET B$=MID$(CHAINE$,B,1)
LET N=0
IF A$=B$ THEN N=N+1 ELSE N=N
LET A= A-X
LET B= B+X
LOOP UNTIL A=1 AND B=L
PRINT N
END
```

Les mêmes erreurs se produisent à l'exécution (du fait que l'initialisation est dans le corps de boucle). Je demande de bien distinguer "les instructions qui doivent être dans le corps de boucle, et celles qui ne sont effectuées qu'une fois". Après beaucoup d'hésitations, les élèves sortent l'initialisation `LET N=0` de la boucle. Le programme donne alors des résultats qui ne sont pas systématiquement zéro ou 1, mais qui cependant diffèrent des résultats attendus; en effet , depuis l'introduction en début de boucle des instructions de calcul des caractères, et de l'instruction alternative, la séquentialité dans le corps de boucle n'est plus cohérente avec l'initialisation et la condition d'arrêt. Les élèves passent les dix minutes restantes à tenter d'interpréter l'erreur, et de corriger le programme, mais sans résultat.

Séance 6: 12 avril; durée 1h; résolution de INSEE1

(il s'agit de vérifier que le premier chiffre d'une chaîne composée de 13 chiffres est 1 ou 2 et que les 3ème et 4ième chiffres forment le numéro d'un mois de l'année, et d'afficher le numéro du département de naissance).

La séance commence par 1/4 h d'explications de ma part concernant la numérotation INSEE; les élèves déclarent ne pas connaître cette numérotation. J'explique également le début de programme présent sur la disquette sous le nom `INSEE.BAS` (voir présentation des problèmes en annexe 9) . Je demande pourquoi le numéro INSEE apparaît comme valeur d'une variable chaîne `INSEE$` , plutôt que d'une variable numérique; seule **Karine** donne un début de réponse, en montrant sur la liste du programme la ligne: `LET INSEE$=MID$(LISTE$,I,13)`, ce qui peut s'interpréter comme l'idée que le numéro INSEE est une chaîne car il est obtenu comme résultat de la

fonction MID$. J'indique par ailleurs qu'un nombre à 13 chiffres ne pourrait être conservé de façon exacte dans une variable numérique.

Natacha Karine: le premier programme obtenu est:

```
LET A$ = MID$(INSEE$,1,1)
IF A$ > 2 THEN PRINT "ERREUR"
LET B$ = MID$(INSEE$,4,2)
IF B$ > 12 OR B$ = 00 THEN PRINT "ERREUR dans les 4 et 5
chiffres"
LET C$=MID$(INSEE$,6,2)
PRINT C$
END
```

Les élèves obtiennent l'erreur de compilation suivante à la deuxième ligne: "*opérande chaîne attendu dans une expression chaîne*"; elles appellent le professeur et le dialogue suivant s'engage:

Professeur: supposez que j'ai plusieurs fois à écrire BONJOUR, que vous ne vouliez pas écrire BONJOUR à chaque fois, vous prendrez une variable dans laquelle je vais ranger BONJOUR; ça ira plus vite j'écrirai PRINT A$; quelle syntaxe... comment je vais l'écrire ? donc LET A$= , qu'est ce que j'écris derrière ?

Elèves: BONJOUR

Professeur: comme ça ? écrivez le moi ...entre..

Elèves: entre crochets

Professeur: c'est pas des crochets ...

Elèves: guillemets

Professeur: guillemets: quand on veut donner une chaîne de caractères, on la mets entre guillemets; si tu veux que ce soit la chaîne de caractères 2, il faut la mettre entre guillemets... .

Ensuite , les lignes 2 et 5 du programme sont corrigées: les chaînes constantes constituées de chiffres sont mises entre guillemets:

```
IF A$="23"     (ligne 2)
IF B$ > "12" OR B$ = "00" (ligne 5)
```

Ces modifications conduisent au succès; le professeur et moi-même attirons alors l'attention des élèves sur la signification de l'inégalité B$ > "12". Nous tentons de leur montrer par des exemples que ce n'est pas l'inégalité dans les entiers qui intervient ici (ce que semblent penser les élèves), mais l'ordre alphanumérique (conçu comme une extension de l'ordre alphabétique à tous les caractères définis pour l'ordinateur, y compris les chiffres et les signes de ponctuation; exemple: "chaton" < "souris") , ce qui conduit le professeur à parler du codage ASCII.

Jérôme (**Arnaud** est absent) le premier programme est écrit:

```
LET A = MID$(INSEE$,1,1)
IF A > 2 THEN PRINT "ERREUR"
LET B = MID$(INSEE$,4,2)
IF B > 12 THEN PRINT "ERREUR dans les 4 et 5 chiffres"
```

L'élève obtient l'erreur de compilation "*opérateur de comparaison attendu dans une expression numérique*" à la première ligne; cette erreur s'explique par le fait que A

étant une variable numérique, l'expression qui suit le signe = de l'affectation doit être numérique ou booléenne; une expression commençant par une chaîne ne peut être numérique; le compilateur attend donc un opérateur booléen constituant la suite d'une expression booléenne commençant par la chaîne. Le diagnostic porté par le message d'erreur est sans rapport avec l'erreur de l'élève: il s'agit ici d'une erreur de déclaration de type;la variable A est déclarée implicitement numérique, alors qu'elle devrait être déclarée de type chaîne.

J'explique à l'élève que si A est résultat d'une fonction MID$, ce doit être une variable chaîne. L'élève corrige les trois premières lignes (A, B, C remplacés par A$, B$, C$, et guillemets autour des chaînes formées de chiffres.

```
LET A$ = MID$(INSEE$,1,1)
IF A$ ="1" OR A$="2" THEN PRINT "BIEN REPONDU" ELSE PRINT
"ERREUR"
LET B$ = MID$(INSEE$,4,2)
IF B$ > "12" THEN PRINT "ERREUR dans les 4 et 5 chiffres"
```

Pour la quatrième ligne, l'élève hésite, et demande si l'expression IF B$ > "12" est possible. Je tente de lui expliquer qu'une telle expression est possible, mais qu'elle n'a pas la signification de la comparaison des expressions numériques, en prenant les mêmes exemples qu'avec **Karine** et **Natacha**. Par la suite, je lui fais remarquer que le cas erroné où les deux chiffres constituants du mois de naissance seraient 00, n'est pas pris en compte; la quatrième ligne est corrigée en:

```
IF B$ > "12" OR B$="00" THEN ...
```

Séance 7: 17 avril; durée:1h30; résolution de INSEE2 et INSEE3

(dans **INSEE2**, les élèves doivent calculer l'age au premier janvier 89 du titulaire d'un numéro INSEE; ils prennent connaissance de la fonction de conversion de type VAL, en exécutant un programme appelé TESTVAL; dans **INSEE3**, ils ont à former le numéro INSEE d'une personne après avoir entré les renseignements le concernant.

Karine Natacha

Les élèves chargent et exécutent le programme TESTVAL, comme l'énoncé les y invite; Elles entrent 6 pour A$; je leur indique d'entrer un nombre plus important pour B$; elles entrent 786 pour B$; à la question "que vaut A$ + B$" **Natacha** propose d'entrer 792 et **Karine** 6786; elles se décident finalement pour 6786 , ce qui est la réponse correcte; à la question "que vaut A + B", elles entrent 786 (réponse correcte). Pour la résolution de **INSEE2**, le premier programme, obtenu de façon spontanée est:

```
LET D$=MID$(INSEE$,2,2)
89-D=VAL(D$)
PRINT D
```

ce qui entraîne l'erreur de compilation: "*INSTRUCTION ATTENDUE*" à la 3ème ligne (le compilateur considère 89 comme un numéro de ligne; il attend donc la première instruction de cette ligne). Je demande quelle est la signification de la ligne: **Karine** répond que "on a D$ et ça va le transformer en un nombre". Je demande quelle est la signification de l'égalité, et je demande de comparer à l'instruction de la ligne précédente. La réponse de **Karine** est "alors il faut faire deux lignes"; j'attire

l'attention sur la syntaxe de l'affectation . Après plusieurs minutes de réflexion, la ligne est corrigée en:

```
D=VAL(D$)
E=89-D
PRINT E
```

Les élèves se livrent à des essais, et semblent satisfaites; je demande de continuer les essais: les numéros INSEE étant tirés au hasard par le module proposé en début de programme, je souhaite qu'un tirage produise le numéro dont les deux chiffres correspondant à l'année de naissance forment un nombre supérieur à 89; ce cas finit par se produire, et le programme donne comme résultat -9. Les élèves insèrent en avant dernière ligne:

```
IF D$ > "89" THEN 100 + E
```

La compilation conduit à l'erreur: "*Référence à une étiquette ou à un numéro de ligne non défini*"; en effet, le compilateur considère le nombre 100 comme un numéro de ligne (qui n'existe pas ici). Je pose la question "vous ne vouliez pas l'envoyer (le programme) à la ligne 100 ?" Réponse "Non, on voulait lui ajouter 100" J'indique qu'il s'agit de "modifier E, donc de lui donner une valeur " **Natacha** répond que "E a déjà une valeur" . J'indique qu'il s'agit de lui donner une nouvelle valeur, et je fais le lien avec l'incrémentation d'un compteur. Il faudra plusieurs minutes de réflexion, et un nouveau dialogue avec le professeur pour que les élèves corrigent en:

```
IF D$ > "89" THEN E=100 + E
```

Le professeur fait ensuite remarquer qu'il y a une erreur sur le calcul de l'age (une personne née en 88 aurait un an au premier janvier 89); pour leur faire comprendre l'erreur, nous introduisons un numéro INSEE avec l'année de naissance des élèves (73); La réponse est 16, alors que les élèves reconnaissent être âgées de 15 ans; elles changent la ligne `E=89-D`, en `E=88 - D`; mais les élèves gardent un doute à cause de personnes qui seraient nées en janvier, février ou mars; nous attirons l'attention sur l'énoncé: "age au premier janvier".

INSEE3: Le premier programme écrit est le suivant:

```
CLS
PRINT "SEXE = F OU M "
INPUT SEXE$
IF SEXE$=F$ THEN INSEE$=2$
IF SEXE$=M$ THEN INSEE$=1$
```

La compilation donne une erreur "*identificateur inconnu*" en troisième ligne. Les deux dernières lignes sont corrigées en:

```
IF SEXE$=F$ THEN INSEE$="2"
IF SEXE$=M$ THEN INSEE$="1"
```

Puis les lignes suivantes sont ajoutées:

```
PRINT "ANNEE ET MOIS DE NAISSANCE SANS ESPACE"
INPUT A$
PRINT "DEPARTEMENT DE NAISSANCE"
INPUT B$
```

Suite à une discussion entre elles, les élèves changent les identificateurs, et obtiennent le programme suivant:

```
CLS
PRINT "SEXE = F OU M "
INPUT SEXE$
IF SEXE$=F$ THEN A$="2"
IF SEXE$=M$ THEN A$="1"
PRINT "ANNEE ET MOIS DE NAISSANCE SANS ESPACE"
INPUT B$
PRINT "DEPARTEMENT DE NAISSANCE"
INPUT C$
INSEE$=A$ + B$ + C$
```

puis une avant-dernière ligne est ajoutée: `LET D$=XXXXXX` et la dernière ligne devient:

```
INSEE$=A$ + B$ + C$ + D$
```

Elles obtiennent un erreur de compilation "*opérande chaîne attendu dans une expression chaîne*", à l'avant dernière ligne; elles corrigent cette ligne en: `LET D$="XXXXXX"` et ajoutent une dernière ligne `PRINT INSEE$` A ce stade, le programme est correct, à l'exception des 3ème et 4ème lignes où les expressions `F$` et `M$` sont comprises du compilateur comme des identificateurs de variables chaînes; ces variables n'ayant pas reçu d'affectation, le compilateur ne signale pas d'erreur et leur attribue la valeur "chaîne vide"; ainsi les deux conditions des lignes trois et quatre ne sont pas vérifiées, et `A$` se voit attribuer la valeur "chaîne vide", et donc le premier chiffre qui repère le Sexe est absent du numéro.

J'attire leur attention sur la signification du "`$`" dans les identificateurs de chaînes, et sur la différence entre identificateurs et constantes, et les élèves corrigent les deux lignes en question en:

```
IF SEXE$="F" THEN A$="2"
IF SEXE$="M" THEN A$="1"
```

Jérôme Arnaud: Le programme TESTVAL est chargé est exécuté; la valeur `1` est entrée pour `A$`, la valeur `72` pour `B$`; les réponses aux questions sont correctes (`172` pour `A$+B$`, `73` pour `A+B`). Le programme est écrit de la façon suivante:

```
LET A$=MID$(INSEE$,2,2)
LET A=VAL(A$)
REM B= Age du titulaire
LET B=89-A
PRINT B
```

Je fais faire les mêmes essais que pour l'autre groupe, de façon à faire apparaître un résultat négatif; une avant dernière ligne est commencée sous la forme:

```
IF A>89 THEN...
```

Les élèves hésitent très longtemps sur la fin de la ligne; des graphismes sur brouillons tels que

```
90 ... 99 ans
```

```
91 ... 98 ans
```

témoignent de leur recherche. Je leur fait remarquer que 89 représente en fait 1989, et qu'une date de naissance telle que 92 représente en fait 1892 . et la ligne est terminée en:

```
IF A>89 THEN B=B + 100
```

Je fais ensuite la même remarque qu'à l'autre groupe sur l'erreur dans le calcul de l'age; les élèves corrigent d'abord l'avant dernière ligne en:

```
IF A>89 THEN B=(B + 100) - 1
```

Cette modification n'a pas d'influence sur le résultat dans les cas usuels, qui sont ceux des essais que font les élèves; ils reconnaissent qu'il convient de modifier la 4ème ligne en ligne en: `LET B= (89 - A) -1`, et rétablissent la dernière ligne.

INSEE3 Le premier programme s'écrit:

```
PRINT "Quel est votre sexe ?"
INPUT Z
PRINT "Quel est votre mois de naissance ?"
INPUT X
PRINT "Quel est votre année de naissance ?"
INPUT Y
PRINT "Quel est votre département de naissance ?"
INPUT K
```

Les élèves obtiennent une erreur à l'exécution de la seconde ligne, et transforment les identificateur `Z,X,Y,K` en `Z$,X$,Y$,K$`. Le programme n'est pas terminé à la fin de la séance.

Séance 9: 19 avril; durée 1h; suite de la résolution de INSEE3

Jérôme Arnaud

Les élèves ajoutent au programme obtenu à la séance précédente les lignes suivantes:

```
IF Z$="MASCULIN" THEN Z$="1"
IF Z$="FEMININ" THEN Z$="2"
```

et commencent une ligne `IF X$=...` Je demande "pourquoi ce test sur `X$` ?" Il n' y a pas de réponse, mais **Jérôme** change la troisième ligne en:

```
PRINT "Quel est votre mois de naissance EN CHIFFRES ?"
```

je comprends alors que les élèves voulaient faire entrer le mois sous forme d'un libellé en toutes lettres. Puis les élèves ajoutent les lignes suivantes:

```
LET Z=VAL(Z$)
LET X=VAL(X$)
LET Y=VAL(Y$)
LET D=VAL(D$)
PRINT "ZYXKWWWWWW"
```

A l'exécution, après avoir entré leurs propres caractéristiques, les élèves obtiennent évidemment comme résultat `ZYXKWWWWWW` . Je demande "Si j'essaie avec mon année de naissance, etc... est-ce que ça va donner la même chose ?" Réponse de **Arnaud**:" Oui, parce que ça ne va pas " Il montre la dernière ligne. Les lignes: `LET Z=VAL(Z$)...LET D=VAL(D$)` sont supprimées, et la dernière ligne est transformée en: `PRINT "Z$Y$X$K$WWWWWWW"` Nouveau résultat erroné à l'exécution: `Z$Y$X$K$WWWWWWW`

J'indique qu'il y a un mélange de variables et de constantes; je demande de porter son attention sur l'énoncé; l'énoncé demande que "l'ordinateur calcule et affiche le numéro INSEE"; j'insiste sur l'aspect calcul; une variable `INSEE$` doit recevoir une valeur par affectation. Je rappelle également qu'on dispose d'une fonction pour "mettre bout à bout deux chaînes". Après dix autres minutes de recherche, les élèves produisent les lignes suivantes:

```
LET W$="WWWWWW"
LET INSEE$=Z$+Y$+X$+K$+W$
```

ce qui termine le programme de façon correcte.

Karine Natacha Résolution de **FILE**: il s'agit de compléter un programme gérant la file d'attente que constitue la salle d'attente d'un médecin; les élèves ont à compléter deux modules: l'un (correspondant à une action du médecin) doit afficher la lettre-code du patient le plus ancien dans la salle d'attente, et mettre à jour la file; l'autre correspond à l'action d'un patient arrivant dans la salle: il entre sa lettre-code, et l'ordinateur met à jour la file.

Les élèves demandent des explications sur l'énoncé; en fait cet énoncé demande à être expliqué oralement: rôle de chacun des acteurs (médecin, patient, ordinateur), façon dont la chaîne `LISTE$` (qui représente la file) évolue dans le programme. Les élèves ne s'attachent pas à découvrir le fonctionnement des boucles. Elles se concentrent sur l'écriture des modules à compléter. Elles complètent d'abord le module correspondant à l'action du médecin. Le module est d'abord complété en:

```
LISTE$=LISTE$-1
```

Cette ligne a été écrite par **Karine**, mais **Natacha** semble ne pas être d'accord; l'écriture est transformée en `LISTE$=LISTE$-"1"` mais les élèves ne semblent pas satisfaites; je propose un essai de compilation. On obtient l'erreur "*Identificateur inconnu/erreur de syntaxe*". J'explique que si l'opération + a un sens dans les chaînes de caractères, l'opération - n'a de sens que dans les nombres. Les élèves demandent qui doit "donner la lettre du patient?" dans ce module. Je précise qu'ici, c'est l'ordinateur, et que donc il doit y avoir un affichage. Les élèves transforment la ligne en:

```
MID$(LISTE$,1,1)
```

Natacha: "il manque quelque-chose"
Karine: "le nom de la variable"
Je demande "Que doit-on en faire ?"
Natacha: "faut l'enlever"
Je précise: "oui, mais pour l'afficher"
le module est donc transformé en:

```
A$=MID$(LISTE$,1,1)
PRINT A$
```

Les élèves ne semblent pas satisfaites; elles ajoutent l'instruction suivante:

```
LISTE$= LISTE$ - 1 ,
```

mais restent hésitantes.

La question de la mise à jour de la chaîne reste posée pour elle; je leur indique que l'emploi de la fonction `MID$`, avec des arguments adaptés, constitue une solution. Elles effacent cette instruction, et **Karine** écrit: `B$=MID$(A$,L-1,1)`; cette expression est contestée par **Natacha**: "`A$` c'est une seule lettre".

La recherche ne progressant pas, je propose de laisser en attente, et de passer au second module. Ce module est complète d'abord en:

```
PRINT "ENTREZ VOTRE LETTRE"
INPUT LETTRE$
```

Les élèves cherchent ensuite comment "ajouter" `LETTRE$`; je fais référence à **INSEE3** (où les élèves ont eu à constituer une chaîne comme concaténation de chaînes élémentaires). Les élèves hésitent encore longtemps avant d'écrire: `LISTE$=LISTE$+LETTRE$` Je propose un essai, sachant qu'il est clair que le premier module n'est pas terminé. L'essai est concluant (le programme fonctionne, si ce n'est que la liste n'est pas mise à jour après l'entrée du signe "-") . La séance se termine sans que les élèves aient trouvé une solution pour ce premier module.

Séance 9: 24 avril; durée 1h15mn; fin de la résolution de FILE

Natacha (**Karine** est absente): fin de la résolution de **FILE**

A la suite de la séance précédente, il restait à traiter l'élimination du premier caractère de `LISTE$`, dans le bloc exécuté lorsque le médecin fait entrer un patient pour la visite. **Natacha** écrit d'abord la ligne: `LISTE$=LISTE$ - A$`, puis l'efface en disant: "on ne peut pas mettre des - " (dans les chaînes). J'indique qu'en effet le signe - n'a de sens que pour les nombres; que par contre, pour les chaînes, on dispose de la fonction `MID$` qui suffit ici; **Natacha** commence: `LISTE$=MID$(LISTE$,`

Je demande de remplir d'abord le dernier argument, nombre de caractères (plus facile à trouver); l'élève propose `L-1`, puis se reprend, en disant que ce n'est pas possible dans les chaînes; j'indique qu'ici - a un sens puisqu'il s'agit de nombres. Le second argument (rang à partir duquel on prend la sous-chaîne) est plus difficile à trouver; l'élève propose à plusieurs reprises le nombre `1`, en donnant comme raison que c'est la première lettre qu'on veut retirer; il faudra un essai d'exécution pour que l'élève convienne que `MID$(LISTE$,1,L-1)` est la chaîne sans la dernière lettre, et qu'elle corrige la ligne en `LISTE$ = MID$(LISTE$,2,L-1)`

L'élève fait ensuite un essai d'exécution; il conduit à une erreur d'exécution: "appel de fonction invalide"; en effet, la variable `L` reçoit la valeur zéro en début de programme; pour l'élève, `L` constitue la longueur (nombre de caractères) de la chaîne, mais il n'y a pas de mise à jour de cette variable dans la boucle. Je l'indique à l'élève, et lui demande comment `L` évolue; j'obtiens la réponse "il augmente de un"; constatant que la manière dont `L` évolue n'est pas claire (dans le module concerné, `L` doit diminuer de un), j'indique d'affecter à `L` la longueur de `LISTE$` en début de boucle.

Le programme est maintenant correct, si ce n'est que l'entrée (par le médecin) du signe - conduit à l'erreur "*appel de fonction invalide*" dans le cas où `LISTE$` est vide (cas où il n'y a plus de patients en attente). Les essais conduits par l'élève ne font pas apparaître ce défaut; je lui indique donc un essai qui conduit à cette erreur, et je lui demande de prendre en compte cette question: faire afficher un message "plus de patients" dans ce cas.

L'élève demande si l'expression `IF L=0` a un sens ? ma réponse est positive; l'élève écrit donc la ligne: `IF L=0 THEN PRINT "PLUS DE PATIENTS"`; mais elle place cette ligne entre les deux modules, ce qui fait que l'erreur d'exécution demeure dans le cas de l'entrée du signe "-" avec une liste vide. Le professeur met l'accent sur la structure d'alternatives imbriquées du premier module correspondant à l'action du médecin. La réalisation par l'élève de cette structure est assez laborieuse, compliquée

par l'oubli d'un ENDIF. Une heure après le début de la séance, le programme est terminé; le module en question s'écrit:

```
IF L=0 THEN
                PRINT "PLUS DE PATIENTS"
          ELSE
                A$=MID$(LISTE$,1,1)
                PRINT A$
                LISTE$ = MID$(LISTE$,2,L-1)
ENDIF
```

Arnaud-Jérôme: résolution de **FILE.**

Les élèves n'avaient pas commencé à écrire un programme lors de la séance précédente; je relis donc l'énoncé avec eux; j'attire leur attention sur les deux modules à compléter , et leur indique que le second est le plus facile (c'est celui qui correspond à l'arrivée d'un patient dans la salle d'attente); Les élèves complètent ce module en:

```
PRINT tapez une lettre
INPUT G$
```

Puis ils complètent le premier module en: `PRINT MID$(LISTE$,1,1)` Ils reviennent ensuite à l'autre module dont ils corrigent la troisième ligne en: `INPUT LISTE$` Ils me demandent si cette réponse convient; je leur indique qu'ainsi tout ce que contenait `LISTE$` auparavant est perdu.

Jérôme revient à `INPUT G$`, et ajoute la ligne `LISTE$=LISTE$+G$` Les élèves reviennent ensuite au premier module qu'ils complètent en: `LET LISTE$=LISTE$-MID$(LISTE$,1,1)`.

J'indique que "-" n'a pas de sens pour les chaînes; les élèves répondent que "+" a bien un sens ! j'explique que "+" n'a pas le sens de l'addition, et que si "-" avait le sens d'ôter un caractère d'une chaîne, il y aurait une ambiguïté, la chaîne pouvant comporter plusieurs fois le caractère.

J'indique que la fonction `MID$` suffit pour résoudre cette question; **Jérôme** écrit la ligne: `LISTE$ = MID$(LISTE$,2,L-1)` , mais cette ligne n'est pas comprise par **Arnaud** qui répète qu'il s'agit d'enlever le premier caractère; je suggère donc de faire des essais d'exécution, et attire l'attention sur la nécessité de donner une valeur à la variable `L`. A la suite des essais, les élèves insèrent une ligne `LET LEN(LISTE$)`; à la suite d'une erreur de compilation, les élèves la corrigent en: `LET L=LEN(LISTE$)` .

J'indique alors de prendre en compte le cas où la liste des patients en attente est vide alors que le médecin entre le signe "-". Les élèves écrivent une ligne `IF LISTE$="" THEN PRINT "PLUS DE PATIENTS"`, mais ils connaissent les mêmes difficultés que **Natacha** pour l'articuler avec le module `IF REP$="-"`; la fin de la séance étant proche, je leur indique comment rédiger cette articulation.

Le texte de l'épreuve sur papier

question 1 (**ONJOURB**)
Ecris un programme pour que l'utilisateur, ayant entré.une chaîne au clavier, l'ordinateur affiche la chaîne en passant la première lettre à la fin.

Exemple: si l'utilisateur entre "BONJOUR", l'ordinateur affichera "ONJOURB". Complète le programme:

```
PRINT "Entrez votre chaîne"
INPUT CHAINE$
```

question 2 (**SOMME**)
Ecris un programme pour que, l'utilisateur ayant entré un nombre entier au clavier, l'ordinateur affiche la somme du premier et du dernier chiffre.
Exemple: nombre 153 réponse 4

question 3 (**CRYPTAGE**)
Deux correspondants souhaitent échanger des messages confidentiels.
C'est pourquoi ils conviennent du cryptage suivant de leurs message:
- ils conviennent d'une chaîne de caractères comportant les 26 lettres de l'alphabet dans un ordre donné; on prendra ici la chaîne "azertyuiopqsdfghjklmwxcvbna"
- le cryptage consiste à remplacer chaque lettre du message par la lettre qui la suit dans la chaîne.
Ainsi, le message "bonjour" est crypté en "npakpit".
On veut un programme tel que:
- en entrée on donne une lettre de l'alphabet,
- en sortie on obtient la lettre correspondante dans le cryptage.
Complète le programme:

```
LET ALPHABET$ = "azertyuiopqsdfghjklmwxcvbna"
PRINT "Entrez la lettre (minuscule) "
INPUT LETTRE$
.....
.....
PRINT "Voici la lettre codée"
PRINT LETTRE1$
```

Réponses des élèves à l'épreuve sur papier

Julien P.

1ère question:

```
L=LEN(CHAINE$)
PRINT MID$(CHAINE$, 2, L-1) + MID$(CHAINE$,1,1)
```

2 ème question:

```
PRINT "ENTRER UN NOMBRE "
INPUT NB$
L=LEN(NB$)
PRINT VAL( MID$(NB$,1,1)) + VAL (MID$(NB$,L,1))
```

3 ème question:

```
DO
COMP=COMP + 1
A$ = MID$(ALPHABET$,COMP,1)
LOOP UNTIL A$=LETTRE$
LETTRE$=MID$(ALPHABET$,COMP+1,1)
```

Michaël

1ère question:

```
PRINT MID$(CHAINE$,1,1)
L=LEN(CHAINE$)
LET CHAINES=MID$(CHAINE$, 2, L)
```

2 ème question:

```
PRINT "ENTRER UN NOMBRE "
```

```
INPUT X$
LET L=LEN(X$)
LET R$=MID$(X$,1,1)
LET G$=MID$(X$,L,1)
LET R=VAL(R$)
LET G=VAL(G$)
PRINT G + R
END
```

3 ème question:

```
DO
LET L = 1
LET LETTRE1$ = MID$(ALPHABET$,L,1)
LET LETTRE$ = MID$(LETTRE1$,L,-1),
LOOP UNTIL LETTRE$ =LETTRE1$
PRINT LETTRE1$
```

Natacha

1ère question:

```
LET A$=MID$(CHAINE$,1,1)
LET CHAINE$=MID$(CHAINE$, 2, L-1)
LET CHAINE$=CHAINE$ + A$
END
```

2 ème question:

```
PRINT "ENTREZ VOTRE NOMBRE "
INPUT NOMBRE$
LET A$=MID$(NOMBRE$,1,1)
INPUT A$
A=VAL(A$)
LET B$=MID$(NOMBRE$,L,1)
INPUT B$
B=VAL(B$)
LET C = A + B
PRINT C
```

3 ème question:

```
LETTRE1$ = MID$(ALPHABET$,LETTRE$ + 1,1)
PRINT "VOICI LA LETTRE CODEE"
PRINT LETTRE1$
```

Karine

1ère question:

```
LET A$= MID$(chaîne$,1,1)
LET B$= chaîne$ + A$
PRINT B$
```

2 ème question:

```
PRINT "Entrez votre nombre "
INPUT nombre$
LET L=LEN(nombre$): REM longueur du nombre
LET A$ = MID$(nombre$,1,1): rem premier chiffre
A=Val (A$): rem pour transformer la chaîne en un nombre
LET B$= MID$(nombre$,L,1): rem dernier chiffre
B= Val(B$)
C= A + B
PRINT C
```

3 ème question:

```
LET x=x + 1
```

```
LET A$ = MID$(LETTRE$,x,1)
LET B$=MID$(ALPHABET$,x+1,1)
PRINT B$
```

Jérôme

1ère question:

```
LET A$= MID$(CHAINE$,1,1)
LET CHAINE$= CHAINE$ + A$
PRINT CHAINE$
```

2ème question:

```
PRINT "ENTREZ VOTRE NOMBRE "
INPUT NOMBRE$
LET L=LEN(NOMBRE$)
LET A$=MID$(NOMBRE$,1,1)
INPUT A$
A=VAL(A$)
LET B$=MID$(NOMBRE$,L,1)
B=VAL(B$)
LET NOMBRE = A + B
PRINT NOMBRE
```

3 ème question:

```
LET X=0
DO
X=X + 1
LET LETTRE1$ = MID$(ALPHABET$,X,1)
LOOP UNTIL LETTRE1$=LETTRE$
LET LETTRE2$=MID$(ALPHABET$,X+1,1)
PRINT "Voici la lettre codée"
PRINT LETTRE2$
```

Annexes au chapitre 12

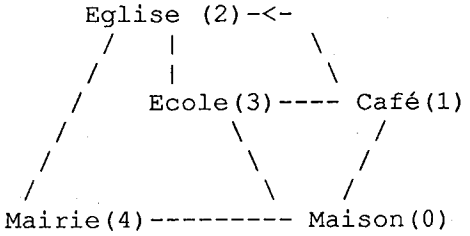
1. Le premier problème (invitation)

J'ai 5 amis : Marie, Marc, Luc, Janine et Jean.
Je souhaite les inviter à dîner, mais il y a des incompatibilités d'humeur et des préférences, que je traduis par les "clauses" suivantes :
- clause 1 : "Marie et Jean ne s'entendent pas : il ne faut pas les inviter ensemble ".
- clause 2 : "Marc et Marie ne viendront pas l'un sans l'autre : si j'invite l'un, il faut que j'invite l'autre".
- clause 3 : "Si j'invite Janine, il faut que j'invite Luc ou Jean, mais on ne peut pas les inviter tous les trois ensemble"

On veut écrire un programme qui permet de savoir si une invitation (c'est-à-dire un groupe d'amis à inviter) est possible :
Ecris un programme qui pose 5 questions "On invite Marie (O/N)" ... puis calcule et affiche la valeur de vérité de chaque clause (c'est-à-dire si la clause est ou non respectée).

2. Le second problème (village)

Ma maison est dans un village qui comprend également un café, une église, une école et une mairie ; il y a des rues ; voici le plan :



La rue du Café à l'Eglise est en sens unique; (comme je circule à bicyclette, il ne m'est pas possible de l'emprunter de l'Eglise vers le Café). Le dimanche, je fais une promenade ; je part de ma maison et j'y reviens après être passé une seule fois en chacun des autres bâtiments marqués sur le plan ; pour m'en rappeler, je note dans l'ordre les points où je suis passé ; par exemple : 1 3 2 4.
Je voudrais un programme qui m'indique si une promenade est possible ; par exemple : 1 2 3 4 , ou 2 1 3 4 ne sont pas possibles.
Ecris un programme où l'utilisateur entre une suite de 4 chiffres désignant une promenade, et où l'ordinateur répond "possible" ou "impossible".

3. Protocoles suite à la recherche sur papier.du premier problème (invitation)

On trouvera ci-dessous une description de chaque protocole. Nous distinguons, pour chaque protocole, la partie "entrée des données", de la partie "calcul des valeurs de vérité des clauses".

Protocole A1 (D. M.)

l'analyse est donnée sous forme d'un programme **Pascal** sans commentaires, et où les variables ne sont pas déclarées :

- entrée des données : elle comporte deux parties :

  partie 1 : entrée des noms, sous forme d'un tableau de chaînes de caractères d'identificateur `NOM`, à l'aide d'une boucle "`tant que...`" . cette partie de l'algorithme permet d'entrer une liste quelconque de noms (d'amis), à l'initiative de l'utilisateur, mais est en dehors du problème puisqu'on peut considérer les noms Marie, Marc, Luc, Janine et Jean comme des données intangibles du problème ; on pourrait bien sûr envisager un problème plus général, où cette entrée des noms aurait un sens, mais alors il faudrait programmer également l'entrée des régles, ce qui n'est pas envisagé par l'élève. Cette partie comporte une erreur (absence d'indice dans l'expression d'un élément du tableau).

  partie 2 : entrée des valeurs de vérité de l'invitation pour chaque nom ; ces valeurs sont entrées sous forme d'éléments d'un tableau, d'identificateur `INVITE`, indexé comme le tableau `NOM` ; l'affectation `INVITE[I]:=1` laisse penser qu'il s'agit d'un tableau de nombres, mais le type précis n'est pas indiqué. Les valeurs sont entrées dans une boucle `FOR` ; en effet, le nombre d'éléments du tableau `NOM` est connu après la première partie du programme ; une réponse est entrée sous forme d'une chaîne de caractères; si cette réponse est la lettre O, alors le nombre 1 est affecté à l'élément courant du tableau `INVITE` ; nous remarquons que dans le cas contraire, cet élément ne reçoit pas de valeur, ce qui constitue une erreur en **Pascal** (selon l'option de compilation choisie, la valeur par défaut peut être aléatoire, ou une erreur peut être générée lors de l'utilisation de cet élément dans une expression).

- calcul des réponses : il n'y a pas vraiment de calcul, mais affichage conditionnel de messages tels que "`Clause.. respectée`", ou "`Clause... non respectée`". La Clause 1 est correctement traitée, par contre, la Clause 2 est erronée (pas de prise en compte du cas où ni Marc ni Marie ne sont invités parmi les cas où la clause est valide). La clause 3 n'est pas abordée.

Protocole A2 (Laurent)

Il y a l'amorce d'un algorithme ("entrer les noms, boucle 5...") avec un langage algorithmique, et un tableau carré indexé sur les noms des 5 amis : la diagonale est hachurée, et certaines cases comportent l'indication "vrai" ou l'indication "faux". On peut penser que l'élève envisage les couples selon qu'ils soient ou non possibles, compte tenu des clauses : ainsi (Marie, Marc) reçoit la valeur "vrai" (interprétation de la Clause 2) et (Jean,Marie) reçoit la valeur "faux" (interprétation de la Clause 1).

Ce tableau constitue une tentative de représentation des clauses sous forme d'un tableau de booléens ; cette structure peut rendre compte de clauses comme la Clause 1 : la valeur "Faux" attribuée au couple (Jean,Marie) suffit à indiquer qu'une suite d'invitations comportant Jean et Marie ne respecte pas la Clause 1 ; mais elle est impuissante à représenter les clauses 2 et 3 : pour la clause 2, il faudrait attribuer la valeur "Vrai" au couple (Marie, Marc), mais comment repérer par exemple que la clause est respectée si ni Marie ni Marc ne sont invités ?

Protocole A3 (Jérôme)

- L'algorithme : c'est une écriture mêlant des mots en français, et des signes (flèche , égalité) Le calcul est compris comme 4 alternatives (si... ->...) ; la partie condition de cette alternative utilise l'égalité, mais dans le sens non informatique ("invités = Marie et Jean") ; le résultat est une valeur (Faux), suivi d'une action (Fin).
- Le programme : les valeurs de vérité des 5 invitations sont entrées dans une boucle `FOR`, et comme éléments d'un tableau d'identificateur `N` ; `N` est déclaré comme "`integer`", ce qui constitue une erreur (puisqu'il s'agit d'un tableau), mais indique bien que pour l'élève, les éléments du tableau sont des nombres, et non des booléens.

  La partie calcul se compose d'alternatives :

  - une alternative pour la clause 1
  - deux fois deux alternatives en chaîne pour la clause2
  - deux fois trois alternatives en chaîne pour la clause3

la valeur de vérité des clauses est conservée dans une variable ; en cas d'impossibilité, la clause reçoit la valeur 0 . On trouve la même erreur que dans le protocole 1 (pas de valeur affectée aux variables représentatives de la valeur de vérité des clauses dans le cas où la clause est vérifiée). L'utilisation d'alternatives en chaîne (`IF...THEN IF...`) est cependant suffisamment maîtrisée, pour que, sous réserve d'une initialisation des variables représentatives de la valeur de vérité des clauses, le programme soit correct.

Protocole A4 (Hervé , Jean-Manuel, Valérie)

Le protocole est constitué d'un programme **Pascal** très peu commenté. Les valeurs de vérité des noms sont conservées dans des variables dont les identificateurs sont les prénoms correspondants (`MARIE...`) , ce qui constitue un progrès en lisibilité par rapport aux tableaux indexés sur des entiers, utilisés dans les autres protocoles. Une seule variable dont l'identificateur est `POSSIBLE`, représente la possibilité ou l'impossibilité pour l'ensemble des clauses.

Ces variables ont d'abord été déclarées "`Boolean`", en conformité avec la nature des données. Mais ensuite, la mention "`Boolean`" a été rayée et remplacée par la mention "`integer`". C'est l'indice d'une difficulté rencontrée par cet élève dans l'utilisation des booléens. La variable résultat d'identificateur `POSSIBLE` est correctement initialisée ; le calcul se compose d'une alternative par clause ; dans la partie condition de ces alternatives, le connecteur logique "`and`" est utilisé ; pour la clause 2, le connecteur "`<>`" (qui signifie "différent de") est utilisé, ce qui conduit à une écriture particulièrement concise de la condition. On peut remarquer que ce connecteur peut s'utiliser de façon indépendante du type des variables qui représentent la valeur de vérité des invitations (l'utilisation de booléens pour les variables `MARIE..`, ne changerait pas l'écriture). Pour la clause 4, l'absence de décomposition du calcul conduit à l'oubli d'un cas d'impossibilité (le cas où Janine, Luc et Jean sont présents simultanément).

En dehors de cette erreur, le protocole montre une bonne connaissance des connecteurs logiques et de leur utilisation. La difficulté qui a conduit à écarter les booléens au profit des entiers, comme type pour les variables `MARIE...` viendrait donc ici plutôt de l'impossibilité d'entrer une valeur booléenne au clavier.

Protocole A5 (Denis)

L'algorithme est ici un texte un français, écrit sous forme d'une suite d'actions :

- les noms sont entrés au clavier, ce qui, comme dans le protocole 1 doit être considère comme hors sujet.
- la partie calcul est une paraphrase de l'énoncé, précédée de la mention "on vérifie" sans indication sur la façon dont la machine peut procéder pour cette vérification.

Le type des données est précisé : un "tableau de caractères", ou sans doute plutôt un tableau de chaînes, pour les noms ; les booléens sont envisagés pour les "choix d'invitation", sans indication d'une structure qui permettrait de les regrouper. On peut penser qu'ici, le choix des booléens a été fait en conformité avec la nature des données ; l'élève ne s'étant pas confronté à l'écriture d'un algorithme ou d'un programme où l'entrée des données et le calcul auraient été plus précisément décrits, n'a pas eu à remettre en cause ce choix.

Protocole A6 (Carole)

Ce protocole est constitué d'un programme **Pascal** non commenté, comportant seulement le calcul de la clause 1. Les valeurs de vérité des noms sont conservées dans des variables dont les identificateurs sont les prénoms correspondants (`MARIE...`) ; le problème de l'entrée des valeurs de ces variables n'est pas résolu.

Il n'y a pas calcul d'une valeur pour la clause 1, mais simple affichage d'un message en cas d'impossibilité. Cette partie du programme est une alternative,dont la partie condition utilise le connecteur logique "and" ; les variables booléennes sont traitées comme des types quelconques pour l'écriture de la condition : "`Marie = true and Jean = true`", alors que le type booléen permet une écriture plus concise : "`Marie and Jean`".

Protocole A7 (Alain)

Ce protocole est constitué d'un programme **Pascal** partiellement rédigé, sans commentaires, et sans la partie déclaration de variable ; on comprend cependant que l'élève emploie un tableau de chaînes (identificateur `chaîne[i]`) pour les 5 noms, et un tableau de nombres (identificateur `invite[i]`) pour les valeurs de vérité des invitations, avec une certaine confusion entre les deux.

L'entrée des valeurs de vérité des 5 invitations se fait dans une boucle `FOR...`

Les éléments du tableau sont correctement initialisés à 0 ou à 1, suivant la réponse de l'utilisateur. La forme choisie (entrée dans une variable chaîne de la réponse de l'utilisateur, puis calcul de la valeur de `invite[i]`) permettrait sans difficulté l'utilisation des booléens (puisque `invite[i]` n'est pas entré au clavier).

Le calcul de chacune des clauses se fait par une alternative, mais il n'est pas possible de savoir si l'élève envisage un véritable calcul, avec affectation à une variable, ou un simple affichage, car les secondes parties des alternatives ne sont pas rédigées. La partie "condition" des alternatives utilise des connecteurs généraux (différent de), logiques (`and`) et même numériques : `invite[3] + invite[5] = 2` traduit la condition :"Luc et Jean sont invités" !

Il semble qu'ici l'élève soit à même de résoudre correctement la question de l'entrée de données booléennes, mais qu'une connaissance insuffisante des connecteurs logiques le conduise à préférer représenter les données par des entiers.

4. Protocoles suite à la recherche du premier problème (invitation) en travail dirigé avec ordinateurs

Nous donnons les deux versions données par chacun des élèves (ou groupes d'élèves), et une brève analyse de ces versions. ; compte-tenu d'absences à l'une des séances, certains élèves ont rendu une seule des deux versions.

Protocole B1 (Antony)

Présent seulement à la dernière séance, cet élève emploie les booléens comme l'ensemble de la classe suite à nos sollicitations.

Réalisant trois affichages conditionnels, le programme qu'il produit est de ce point de vue proche des productions spontanées. Par contre, les connecteurs logiques sont employés pour l'écriture des parties conditions. Ces conditions sont les négations des conditions de l'énoncé (de façon qu'un message "non respecté" soit affiché si la condition est remplie).

calcul de **clause 1**:
```
if BOOL[1] and BOOL[5] then writeln...
```
calcul de **clause 2**:
```
if not (BOOL[1]=BOOL[2]) then writeln...
```
calcul de **clause 3**:
```
if not (BOOL[4]=BOOL[3]) or (BOOL[4]=BOOL[5])
                and (BOOL[3]=BOOL[5]) then writeln...
```

Les clauses 1 et 2 sont correctes; la condition de la troisième alternative est vraie dans le cas où Janine est absente, et Luc et Jean présents, ce qui constitue une erreur.

Protocole B2 (Jean-Manuel, Valérie)

(la recherche de ce groupe a donné lieu à un entretien enregistré sur magnétophone rapporté plus loin).

version 1 : absente.

version 2 :

calcul de **clause 1**:
```
(Marie and (not Jean)) or (Jean and (not Marie)) or ((not
Marie) and (not Jean))
```

calcul de **clause 2**:
```
(Marie and Marc) or ((not Marie) and (not Marc))
```

calcul de **clause 3**:
```
c31 := Janine and Jean;
c32 := Janine and Luc;
c33 := (not Janine) and (not Luc) and (not Jean);
clause3:=c31 or c32 or c33;
```

On remarquera dans ce calcul l'emploi systématique d'une forme disjonctive. Il en résulte une expression assez lourde de la clause 1 et de la clause 2 ; le calcul de clause 3 comporte deux erreurs (le cas où Janine, Jean et Luc sont présents simultanément n'est pas exclu, et le cas où Jean et Luc sont présents sans Janine n'est pas autorisé).

Protocole B3 (Jérôme)

(la recherche de cet élève a donné lieu à un entretien enregistré sur magnétophone rapporté plus loin).

version 1 : elle est conforme à l'analyse produite par cet élève lors de la première phase (protocole A3) : emploi d'un type numérique pour les quantités logiques ; emploi du remplacement conditionnel et d'alternatives en chaîne (`IF..THEN IF...`) pour éviter le recours aux connecteurs logiques. L'erreur (absence d'initialisation des résultats) a été corrigée; Le programme est correct, mais difficile à valider.

version 2 : les valeurs de vérité des invitations sont affectées aux éléments d'un tableau de booléens : `n` ; ainsi `n[1]` est `"true"` si Marie est invitée, `"false"` dans le cas contraire.

calcul de **clause 1**:

```
ca:= not( n[1] and n[5]);
```

calcul de **clause 2**:

```
cb:= (n[1] = n[2]);
```

calcul de **clause 3**:

```
cc1:=not(n[4] and n[5] and n[3]);
cc2:=not(n[4] and (not n[5] and (not n[3])));
cc:=cc1 and cc2 ;
```

L'élève traduit ici la définition des clauses selon l'énoncé de façon assez directe ; le parenthésage est largement utilisé pour éviter le recours au connecteur disjonctif (`or`) ; l'élève maîtrise par contre les connecteurs `and` et `not`.

Protocole B4 (Hervé)

version 1 : elle est conforme à l'analyse produite par cet élève lors de la première phase (protocole A4) : emploi d'un type numérique pour les quantités logiques ; calcul d'une valeur de vérité ; emploi du connecteur logique `"and"`, et de l'égalité et de l'inégalité. L'erreur dans l'expression de la clause 3 a été corrigée, et une expression particulièrement concise a été obtenue pour la négation de cette clause: `(janine = 1) and (luc = jean)`.

version 2 : absente.

Protocole B5 (Denis)

version 1 : l'analyse produite sur papier (protocole A5) était une simple paraphrase de l'énoncé, énonçant cependant que "les choix d'invitation sont des booléens". Ici, ces "choix d'invitation" sont affectés aux éléments d'un tableau d'entiers, et la valeur de vérité de chacune des clauses est affectée également à une variable numérique ; le calcul se décompose en 4 alternatives de la forme : `if (c[...]=1 and c[...]=1) then ...:=0` Le programme prend donc en compte quatre exclusions mutuelles ; l'exclusion mutuelle, comme indiqué plus haut, permet de décrire la clause 1, mais est impuissant à décrire les autres clauses.

version 2 : absente

Protocole B6 (Carole)

version 1 : absente.

version 2 :

calcul de **clause 1**:
```
 Not( Marie and Jean)
```
calcul de **clause 2**:
```
 Marc = Marie.
```
calcul de **clause 3**: `non fait.`

Protocole B7 (Alain)

version 1 : elle est conforme à l'analyse produite par cet élève lors de la première phase (protocole A7) : emploi d'un type numérique pour les quantités logiques ; emploi d'opérations numériques dans la partie condition des alternatives (`invitation[3]+invitation[5]=2` pour Luc et Marc) ; il n'y pas de calcul de valeur de vérité des clauses, mais simple affichage conditionnel.

version 2 :

calcul de **clause 1**:
```
 clause1:=not (invitation[1] and invitation[5])
```
calcul de **clause 2**:
```
 clause2:=not (invitation[1]<>invitation[2])
```
calcul de **clause 3**:
```
clause3:=not (invitation[4] and
(invitation[3]=invitation[5]));
```

Les clauses 1 et 2 sont des traductions directes de l'énoncé, sans même la simplification évidente de clause2 ; par contre, l'élève fait preuve d'invention pour une construction particulièrement concise de clause3 (probablement inspirée de la version 1 du procole B4).

5. Les difficultés à utiliser les booléens (entretien avec Jérôme suite au premier problème invitation)

Les conditions de déroulement de l'entretien sont les mêmes qu'en annexe 6 paragraphes **3** et **4**: l'élève est devant l'ordinateur, avec le résultat de sa recherche écrite ; nous enregistrons au magnétophone, prenons des notes et relançons la recherche. Le dialogue enregistré est à droite, les commentaires à gauche. Nos interventions sont en italiques . Les remarques de l'élève sont précédées de son initiale (**J.**).

La première phase porte sur la version 1 du programme écrite par **Jérôme** ; cette version utilise des entiers pour les valeurs logiques, et beaucoup d'alternatives ; elle est conforme à la recherche sur papier de l'élève (Protocole A3). La phase consiste à faire des tests d'exécution du programme ; il y a une erreur de recopie qui est détectée par un test et à corrigée.

• *Essaie d'inviter Janine et Jean.*

J. Janine et Jean

• *avec d'autres aussi*

J. Marie et Jean par exemple ...

J. clause1 : 0

• *Oui clause 1 pas respectée parce que...*

J. C'est bon

• *Essaie d'inviter pas Janine, mais Luc et Jean... Ah non tu invite Janine aussi...*

J. Là c'est pas normal !

• *Oui c'est pas normal qu'on puisse inviter Luc, Janine et Jean en même temps ... Il y a un problème, essaie d'inviter sans Janine, Luc et Jean. La même chose, mais sans Janine*

J. Là c'est bon

• *Là ça ne me paraît pas être cela.*

J. Ah non c'est quand on invite Janine qu'il faut inviter Luc ... Ah oui d'accord...

• si Janine n'est pas invitée, on peut très bien les inviter tous les deux...

J. Ah oui d'accord...

• *Il y a quand même un problème, c'est que...*

J. quand on invitait les trois personnes, ça n'était pas exclu.

J. En fait, si j'invite Janine et Luc...

• *La ligne marquée par le curseur* IF n[4]=1 then if n[3]=0... *Et la ligne suivante, qu'est ce que tu veux dire...*

J. Ah non, je me suis trompé. C'est une faute de frappe (cc:=1 au lieu de cc:=0) J'avais bien marqué ça (la ligne est exacte sur la feuille manuscrite). Je voulais dire si on les invite tous les trois, ça va pas... Je corrige.

• Essaie de les inviter tous sans Jean...

J. pas de problème puisque on

n'invite pas Jean, on peut inviter Marie, on invite Luc, on n'invite pas Jean, donc on peut inviter Janine...

Invitation à passer à la version 2.

• *En fait on devrait constater que le programme est juste, en le lisant, et pas seulement par des tests...Ce serait plus clair si on employant des booléens*
Par exemple, est-ce-que tu pourrais écrire la première clause sans "IF"?

J. La première clause sans IF ...

• *tu peux employer des booléens...*

J. on ne s'en est servi qu'une seule fois, donc je ne sais pas très bien les employer, c'est pour ça.

L'élève a souhaité faire un vecteur des valeurs logiques entrées ; ainsi `n[1]=1` signifie "Marie invitée", `n[5]=1` signifie "Jean invité".

• `ca` *, il vaut* `0` *ou* `1`*... si je pouvais écrire comme* `ca:=(n[1]=1 and n[5]=1)` *;c'est faux dans certains cas, et juste dans d'autres; ici c'est une valeur logique,* `n[1]=1` *ça peut valoir* `1` *ou* `0` *; mais il faudrait déclarer la variable ca comme...*

J. comme booléen !

• *comme booléen.*

• *on pourrait écrire plus simple :*
`ca:= n[1] and n[5];`

J. ah oui, d'accord, si ca vaut 1 et 1 ,on ...

• *ça supposerait que ceci* `(n[1] )` *soit un booléen ; c'est vrai que Marie vient et que Jean vient.*

J. donc ils viennent tous les deux

• *... on s'est trompé ; ça , ça serait juste* `ca:= not n[1] et n[c]` *; ils ne doivent pas venir ensemble...*

J. Ah oui, c'est vrai ;

• *ça supposerait que* `ca` *et les* `n[]` *soient des booléens... on pourrait essayer cela, au moins pour la clause 1*

MODIFICATION DU PROGRAMME...

• ca:= … (dicte) not … and … *ça suppose que* ca *soit booléen … on n'a pas à lui donner une valeur initiale… puisqu'on lui donne une ici…*

J. donc j'efface ici…

• *oui efface…*n *c'est pareil, il faut le déclarer en booléen, puisque ici tu.. il y encore un problème, ici on va faire* readln(n[i]) *; ce n'est pas un problème quand c'est des entiers, mais, par contre c'est un problème quand c'est des booléens ; on va entrer un entier qu'on va appeler* REP, *pour réponse, on dira que* n[i] *a la valeur … pas de* REP, *car* REP *est un entier, et* n[i] *un booléen…donc on peut pas le…on sait que la valeur de* n[i] *est soit* VRAI *si* REP
Il n'est pas possible d'entrer des booléens au clavier.

J. est égal à 1…

• *est égal à 1 d'accord.. et qu'elle soit…*

J. fausse quand…

• *donc il suffit d'écrire* n[i]:= *entre parenthèses* REP= 1.. *Si* REP =1 *la valeur sera bien* VRAI

J. Oui,

• *et si* REP=0, *la valeur sera bien fausse…*

MODIFICATION DU PROGRAMME...

• *donc* n[i]:=... *On l'a pas déclaré* REP…

EXECUTION...

• *le seul problème, c'est qu'il nous envoie* FALSE *; Mais c'est aussi bien que* 0 *ou* 1 *finalement.*

J. c'est sûr , parce que là, on peut pas savoir...

• *on peut peut-être essayer de construire* clause2 *avec les booléens aussi...*

J. ...

• *parce que* n , *il vaut* VRAI *ou* FAUX, n[3] *par exemple, soit c'est* VRAI, *c'est* TRUE, *soit c'est* FALSE... *donc si tu veux , si tu écris* n[1] = TRUE , *c'est la même chose que* n[1] *; si tu veux le contraire, c'est* NOT ... *ce qui revient à* n[i]=0 *de tout à l'heure, c'est* NOT n[i] *; c'est l'inverse, ça veut dire : la condition n'est pas réalisée...*

J. ah oui d'accord..donc ... IF..

Malgré l'exemple de la construction de CLAUSE1, **Jérôme** est tenté d'employer une alternative pour CLAUSE2.

• *essaye d'analyser le problème : Marc et Marie ne viendront pas l'un sans l'autre.*

J. Quand...

• Quelles sont les situations où la clause est vraie ?

J. La clause est vraie quand ...

• *il y a une façon simple de le dire, ça... soit ils ne viennent pas tous les deux, soit ils viennent tous les deux. S'ils viennent tous les deux, alors les valeurs sont pour Marc, c'est* n[2] *et pour Marie* n[1].. *soit on a* n[2]= VRAI, *soit on a* n[1]=FAUX , *premier cas, deuxième cas,* n[2]=FALSE, *et* n[1]=VRAI...

J. oui... je comprends bien cela, mais pour le mettre sous la forme de...

• *Il y a une façon très très simple de le dire, en une seule instruction...*

J. on n'est pas obligé de les mettre tous les deux, on peut en mettre une seule..

• *oui d'accord, on peut dire,* n[2] and n[1] *; ça ça veut dire qu'ils sont vrais tous les deux ; mais ça ne sera pas le seul cas, il faudra dire* not n[2] and not n[1] *; il y a une façon beaucoup plus simple de le dire :* ici TRUE TRUE, FALSE FALSE...

J. (proposition inaudible)

• *oui, mais il y a plus simple ! ils ont la même valeur dans les deux cas ! comment on peut exprimer cela ?*

J. quand il y en a un qui , qui... on peut voir le cas où ça ne marche pas...

Proposition d'une forme disjonctive pour la négation de la clause.

• oui, oui, c'est plus simple de voir.. les deux cas se ressemblent ; ce ne sont pas exactement les mêmes, mais les deux cas se ressemblent ...ils ont une caractéristique commune.

J. quand celui-ci est vrai , l'autre est vrai.

• *donc on peut écrire* cb:= (Marc=Marie)

MODIFICATION du programme ESSAIS.

J. cb... on va inviter Marc et Marie..

• *d'accord, il reste à faire la clause3, on aura un programme qui n'aura pas un seul* IF.

J. cc:=(n[4]=n[3]) or (n[4]=n[5])

Un peu comme le cas précédent, l'expression traduit Janine=Jean ou Janine=Luc, ce qui n'est pas équivalent à la clause de l'énoncé. Pourtant l'expression marque un progrès en ce sens que l'affectation d'une valeur logique a été utilisée.

• *essayons de prendre tous les cas pour vérifier ;*

J. les cas où elle est fausse ...

• *écris le : elle est fausse si ...*

PROPOSITION de **J.** par écrit : n[4] and not (n[5] and n[3]) ; proposition qui résume: Janine est là et Jean et Luc ne sont pas là tous les deux ; il y a manifestement confusion chez **J.** entre not (A and B) et not A and not B.

• *ça veut dire qu'il y l'un ou l'autre, si tu mets* not(n[5] and n[3]).

J. c'est pas évident...

• *ça serait le cas où Janine serait là , d'accord, Janine est là et* not n[4].. *(Luc n'est pas là).*

J. oui...

• *et ...Jean n'est pas là.* (not n[3]). *ça nous donne la condition inverse, on remettra un* NOT *devant pour dire que... si on veut que ces trois conditions soient vérifiées en même temps, on met des* AND...

J. oui...

• *ce sont des questions de logiques,* not (A and B) *ce n'est pas la même chose que* not A and not B, *contrairement à l'addition et à la multiplication...*

J. ce n'est pas une distributivité...

Les tautologies classiques ne sont pas connues.

• *il n' y a pas de distributivité ; c'est vrai, à condition de remplacer ici le* and *par un* or *; ce sont des questions de logiques...*

J. ici, il faut traduire les deux conditions...

• *on peut employer des variables intermédiaires,* cc1, cc2, *pour...on peut appeler ça* cc1, *ça* cc2, *et après cela, on recombinera tout.*

J. on peut pas les combiner ensemble directement ?

• *si mais ça va faire une expression (énorme) . Toujours, quand on a un gros calcul dans les ...dans les entiers, on essaie de séparer ; il vaut mieux sinon, après il y a des parenthèses, c'est...*

J. oui..

• *on arrive à faire des erreurs...*

... RESOLUTION par **J.** des expressions correctes sont trouvées

• *il y a des problèmes de parenthèses ; il vaut mieux en mettre autour de not; on ne sait pas quelles sont les priorités... on va faire afficher* cc1 *et* cc2 *pour voir...*

J. on peut les inviter tous les trois,

• *par exemple pour voir..*

...EXECUTION...

• cc1, *en fait c'est qu'on les invite tous les trois, et la clause* cc2, *on invite Janine et il faut en inviter aussi l'un des deux,*

J. oui..

• *voire les deux...*

J. oui..

• *pourquoi est qu'on a une valeur pour* cc??? *on l'a pas affecté,*

J. oui, parce-que on ne l'a pas initialisé, donc.. donc...

• *donc il a pris une valeur quelconque...*

PROGRAMME DEFINITIF :

```
ca:= not(n[1] and n[5]);
cb:=(n[1] = n[2]);
cc1:=not( n[4] and n[5] and
n[3]);
cc2:=not( n[4] and (not n[5])
and (not n[3]));
cc:=cc1 and cc2;
```

J. oui, voilà..

• *comment peut-on calculer* clause3 *à partir de,.. à partir de* CC1 *et de...*

J. bien oui, quand l'une des deux est fausse, CC est fausse.

• *il serait plus simple de faire comme au dessus.*

J. je change ?

• *non, non, on va calculer* CC *à partir de* CC1 *et* CC2, *comme quand on fait des calculs intermédiaires, on refait une nouvelle ligne...* CC:= *et puis alors, il faut un calcul maintenant en fonction de* CC1 *et de* CC2.

J. Alors.. il faut que les deux conditions soient vérifiées.

• *Tu les a déjà remis en* NOT . *il faut que la sous-clause soit vérifiée, et la sous-clause soit vérifiée ; il n'y a pas besoin de* NOT.

6. Signification de l'affectation et des calculs sur les booléens
(entretien avec Jean-Manuel et Valérie)

Les conditions de passation de l'entretien sont les mêmes que ci-dessus. Le dialogue enregistré est à droite, les commentaires à gauche. Nos interventions sont en italiques, précédées de "•" **J.M.** . désigne Jean-Manuel **V.** désigne Valérie. **Jean-Manuel** (élève de 1ère S) est atteint d'une maladie des muscles ; il n'écrit pas et ne tape rien au clavier ; c'est **Valérie** qui se charge de ces tâches.

L'entretien débute par la comparaison entre la forme proposée dans le programme à compléter, et les intentions des élèves, exprimées dans l'analyse produite en commun avec **Hervé** (protocole A4)

Ce protocole emploie la forme "remplacement conditionnel", alors que le programme à compléter se caractérise par l'affectation du résultat d'un calcul sur des booléens.

J.M. La question que j'avais posé est "pourquoi il y a := " (affectation à compléter dans le programme proposé sur disquette) ?"

• *et la réponse que tu avais donné ?*

J.M. que...que.. il ne faut pas le faire avec des IF... THEN...ELSE

• *c'est celà, parce que, au départ, tu pensais le faire avec des alternatives..*

J.M. Voilà, c'est ça

• *d'accord..*

J.M. cependant que "untel" est pas là, on peut inviter, euh...

• *d'accord..*

• *il avait mis beaucoup de "IF"... en fait, c'est une première solution, mais c'est difficile à lire*

J.M. donc il faudrait un programme ...

• *voilà, en employant le moins de "IF" possible.*

J.M. une autre façon d'exprimer...

• *oui...*

J.M. l'idée ça serait, donc de.. avec les booléens, vrai ou faux

• *oui...*

J.M. mais, comment dire quand même, ça dépend de la valeur "VRAI" ou "FAUX" d'un booléen ?

• *est-ce que vous avez lu le programme là* (l'énoncé de l'exercice comportant un programme à compléter) *. ?*

J.M. oui... d'accord..

• *est-ce que cette écriture vous est familière?*

J.M. Euh Marie:=..." (rep = 'O') ; oui, il fait le test, est-ce que la réponse est égale à oui ou à non . On avais déjà vu cela quelque part, en fait, est-ce que Marie est là ou non... par rapport à la valeur, tu fais O ou N...

• *Qu'est-ce que ça désigne ici* Marie?

V. C'est le.., c'est le ... c'est le booléen..?

• *c'est une variable booléenne, donc elle peut valoir OUI ou NON*

V. OUI ou NON...

J.M. il fait, rep = 'O' donc, est-ce que la réponse est égale à OUI ou NON ...

• *si j'ai mis "Z", qu'est ce qui va se...* Marie va *prendre...?*

Au cours de cette première phase, 1. essaie de vérifier la compréhension de la notion de variable booléenne, à partir de l'interprétation du fragment de programme : write('Marie ') ; readln(rep) ; Marie :=(rep='o') ; 1. tente d'obtenir cette interprétation en demandant la valeur de la variable Marie pour différentes entrées

J.M. euh...ça je sais pas...ça m'étonnerais qu'il aille comme ça...

V. Mmm...

• *Lisez, là ... si je mets "Z"... Si je mets un O majuscule...là (dans le programme) c'est un o minuscule,* rep= o minuscule...

J.M. ... j'hésite entre deux trucs...à mon avis, ou il le prend faux, tout de suite...

• *oui...*

J.M. ou il dit tiens tac, c'est bon,

• *on ne peut pas le savoir, ici ?*

J.M. à mon avis, il le prend faux...

• *on peut, on peut essayer*

J.M. ben oui...

• *attends, il faudrait faire afficher la valeur de* Marie... *vous savez ça, mettre un commentaire* (en fait des accolades pour que la partie suivante du programme ne soit pas compilée).

J.M. je ne vois pas ce que vous voulez dire...

• *comment peut on connaître la valeur de* Marie?

Jean-Manuel pense d'abord que toute entrée ayant une connotation affirmative donnera la valeur vraie à la variable, et que toute entrée ayant une connotation négative donnera la valeur fausse.

J.M. Normalement, c'est nous qui la donnons...

• *oui d'accord, mais comment est-ce qu'on la connaîtra ? non...nous on donne la valeur de* rep...

J.M. ah oui alors faut mettre Marie= la valeur de rep

• *pas exactement, bon il faudrait* "writeln de Marie, *de la valeur de Marie, pour qu'on connaisse la valeur de* Marie.

J.M. ah, ah ben oui...oui, oui, tout à fait...oui ça y est, j'ai compris, pour qu'il affiche Marie= vrai.

• *oui, il mettra* Vrai, *ou* Faux ?

J.M. il mettra Oui ou Non

• *Il nous cause en Anglais...*

J.M. il va mettre, ben...

• True *or* False..

V. Voilà

• *voilà essayons...*

• *on essayera d'abord un o minuscule, ensuite on essayera d'autres...*

J.M. un o minuscule, il va rien faire...puisqu'on avait mis rep=o...

• *essayons...*

J.M. si le o est minuscule, ça va marcher...

Essai en machine d'une valeur non prévue: "O" (majuscule) est essayé alors que le programme prévoit: Marie:=(rep='o').

• *exécutons...* (En fait, l'essai est fait avec un O majuscule en entrée).

L'ordinateur sort "false" en contradiction avec la connotation positive de l'entrée.

J.M. alors, oui, il faut absolument mettre un o minuscule... autrement dit, il faudrait rajouter une chaîne de caractères...

• *en fait c'est pour simplifier ; en fait c'est pour que vous voyez qu'en fait on peut taper n'importe quoi, au lieu de taper* NON..

J.M. et alors, il prendra ...

• *on réessayer en mettant un o minuscule ?*

J.M. ben, il va mettre VRAI...

• *d'accord... vous voyez, c'est pas* VRAI *ou* FAUX *qu'il met c'est...*

J.M. oui, c'est pareil, je peux vous parler en allemand..

Les élèves expérimentent en même temps le résultat de la sortie d'une valeur booléenne.

• *ça je ne sais pas , ou plutôt j'ai su... d'accord, vous voyez ça ; maintenant, vous essayez de voir comment on peut calculer clause1...*

J.M. faites voir un peu le texte... alors Marie et Jean ne s'entendent pas.. oui ça va être encore des IF THEN ELSE...

...

Jean-Manuel a besoin d'une interprétation particulière de l'affectation quand cette affectation concerne des valeurs booléennes : Marie:=(rep='o') est interprété comme :"Marie correspond au cas où rep='o'". Ce qui indique que sa conception de l'affectation acquise sur d'autres types de données n'est pas suffisante pour une extension au type booléen : "rep='o' " reste pour lui une "condition", et non une expression à valeur booléenne.

J.M. j'ai une idée tout d'un coup ; est-ce que je pourrais interpréter là := comme correspond ?

Notre réponse ne met sans doute pas assez en évidence qu'il s'agit d'une affectation.

• *plutôt comme "a comme valeur"*

J.M. parce que si par exemple, parce que := par exemple,donc on peut simplement mettre rep de Marie = faux ou ...

• *non c'est* Marie, *qui vaut...*

J.M. en dehors du.. non parce-que je ne l'ai pas dans la tête, au niveau du principe...

• *oui...*

J.M. fait voir un peu (le texte) : on dit clause1, Marie et Jean ne s'entendent pas...il faut les inviter...

V. il ne faut pas..

J.M. oui je vais y arriver ; il ne faut pas les inviter ensemble ; est-ce qu'il faut mettre clause1:= ... euh...par exemple, Marie= Vrai , Jean=faux ; et puis OR, ensuite...

Dans l'usage qu'ont fait les élèves jusque là, les connecteurs logiques relient des conditions qui elles-même doivent être formées à l'aide d'opérateurs de comparaison. D'où les écritures Marie= Vrai, Jean=faux ..

• *oui...*

J.M. ou bien Jean = faux, Marie = Vrai... Non : oui en fait...

• *c'est vrai qu'on pourrait l'écrire comme cela, sauf que dire* Marie=..., *on pourrait dire* Marie= false, *ou* Marie = true...

J.M. oui, oui, d'accord, d'accord..

L'abréviation "Marie " pour "Marie= true" et "not Jean" pour "Jean=false" est bien acceptée.

• *dire* Marie= true, *ou dire* Marie, *c'est pareil...*

J.M. bon ben, alors Marie, . . . alors...Marie et puis... not Jean..

• Marie and not Jean, *oui...*

J.M. OR...Jean and not Marie...

.... frappe du programme complété...

• *pour les parenthèses, on en met le maximum, quand on ne sait pas (quelles sont les priorités...)... par exemple autour de* not Jean, *tout d'un bloc, comme ça c'est bien le contraire ... Jean n'est pas là...*

J.M. il faudrait mettre des parenthèses autour de Jean et de Marie...

• *non, autour de* not Jean, *comme ça...pour dire c'est* not Jean, *et pas, c'est* not(Jean or Jean and not Marie)...

J.M. ah oui, d'accord...tu mets not Jean... et puis not Marie ; les parenthèses, c'est là haut, en minuscules...

• *il y a encore quelque chose, ici vous avez le* and, *il fait intervenir deux objets, l'un avant, l'autre après, donc parenthésez l'ensemble de l'expression...*

J.M. comment ?

• *parenthésez l'ensemble de l'expression...*

J.M. un truc avant... avant Marie... à l'origine c'est ce que je voulais faire, mais je n'aurais pas mis de parenthèses là (autour de la seconde partie de l'expression)...

• *oui, ça n'aurait peut-être pas été nécessaire...*

J.M. et puis après jean, une seconde parenthèse

J.M. on verra à écrire ça proprement après.. bon alors compile, oui sauve à tout hasard.

J.M. bon alors compile, c'est bon, pas d'erreur... qu'est ce qu'on a dit que...

V. c'est en minuscules..

J.M. il fallait que Marie soit là, et pas Jean, ou le contraire ; bon, mettons OUI pour Marie, on s'en fiche, bon pour jean...ben là, tu mets rien du tout...ça revient au même ...

le fragment de programme :
```
write('Marie ') ;
readln(rep) ;
Marie :=(rep='o')
```
qui a donné lieu au dialogue de la première phase de cet enregistrement a été bien compris.

V. ...

• *N par exemple, ou y, ce qu'on veut...*

J.M. alors, true , donc ça marche !

• *qu'est-ce qu'il faudrait réessayer ...*

V. ben quand ils sont là tous les deux... ...

• *et si je mettais non à Marie et à Jean ?*

Dans le calcul de la clause 1 (forme disjonctive) le cas où Marie et Jean sont absents simultanément a été oublié. Les valeurs critiques de tests sont apportées par nous

J.M. ben ils vont refaire,..., vas y voir...

... résultat `false`...

• *c'est pas normal, là, parce que si je ne les invite ni l'un l'autre, il n'y a pas de raison qu'ils se disputent...hein, ça devrait être bon...*

V. ben oui...

J.M. c'est un problème de ...

• *il faudrait revoir le programme... je n'ai pas souvenir de la phrase que vous avez copié... parce-que, il n'y a que deux cas dans lesquels la clause est vraie...puisque vous avez un* OR... *c'est soit l'un, soit l'autre, éventuellement les deux...mais elles sont incompatibles...donc c'est soit l'un soit l'autre, mais le cas où ni Marie, ni Jean ne viennent, il n'est pas pris en compte dans votre... comme donnant la valeur vraie*

J.M. peut-être que...

• *oui, vous pouvez le rajouter, ou trouver une expression plus simple, mais vous...*

J.M. not marie and not jean...est ce qu'on peut mettre un autre OR...

• *oui, oui, oui, c'est comme des "plus", vous pouvez...*

L'associativité de la disjonction OR n'est pas connue par Jean-Manuel.

J.M. bon, alors, not Marie and not Jean

• *on peut écrire une expression beaucoup plus simple, mais enfin essayez celle-là déjà..*

J.M. Beaucoup plus simple...?

• *oui, simplement en une seule expression...*

J.M. il y a une traduction pour "ni l'un ni l'autre"..?

• *pas les deux à la fois...je voulais dire pour l'ensemble (de la clause) ; d'accord pour* OR not Marie and not Jean...

nous pensons à la forme négative NOT(Marie AND Jean) alors que **Jean-Manuel** est attaché à une forme disjonctive.

...

• *n'hésite pas à mettre des parenthèses, plus on met de parenthèses, et plus c'est lisible...*

Calcul de CLAUSE2: Jean-Manuel commence l'écriture d'une forme disjonctive.

J.M. maintenant, normalement c'est bon... l'autre clause... attend je vais te la traduire A tous les coups ça marche... Marc et Marie ne viendront pas sans l'autre... Bon alors...Marie and Jean or not Marie and not Jean ... Oh, pardon, je me goure de prénom...

V. Marc...

J.M. Marc and Marie or not Marie and not Marc ... Allez roule...

V. (tape la clause)...

Calcul de CLAUSE3

V. clause3, bon on descend

J.M. elle dit quoi la clause3 ?

V. si j'invite Janine, il faut que j'invite Luc ou Jean, mais on ne peut pas les inviter tous les trois ensemble...

J.M. Aïe...on recommence...

V. (relit le texte)...

J.M. parce-que Luc et Jean ne vont pas ensemble...

V. Voilà, ...donc...c'est Janine...Janine...

J.M. on ne doit pas... c'est une condition... c'est si Janine est invitée..IF..

Le connecteur à employer serait effectivement l'implication, ce qui entraîne une confusion avec la forme SI...ALORS de l'alternative.

• *ce que vous pouvez faire, c'est utiliser des sous-clauses, c'est à dire une clause, et puis faire un* AND *ou un* OR *entre..par exemple ici, votre forme, elle est assez compliquée, deux objets sont séparés par* OR,*; on aurait pu dire :* c… clauseA… C21 *par exemple…*

nous proposons de séparer le calcul de CLAUSE3 à l'aide de l'affectation à des variables intermédiaires.

Cette décomposition simplifie l'écriture des formes disjonctives proposées par les élèves.

V. oui…

• C21= Marie and Jean..

V. oui…

• C22= not Marie and not Marc..

J.M. ah, oui d'accord ; là on peut, clause31..

• *et puis* clause32… *et puis* clause....

J.M. clause31=Janine…

• *oui…*

V. on fait clause3=Janine ?

J.M. clause31= Janine…

V. je le met où, aussi, attend…à l'intérieur d'une apostrophe? si Janine…

J.M. clause31= Janine…

V. on le note comment aussi ?

J.M. j'ai une idée , t'a qu'à mettre C1..

V. à l'intérieur de la clause 3 ?

Une erreur dans l'emploi de l'affectation; les élèves ne feraient sans doute pas cette erreur avec d'autres types de données.

J.M. C1:= ..C31...:= Janine...

V. je mets une parenthèse..?

J.M. non, non y a que Janine, donc c'est bon...

V. c'est pas utile... Janine... oui...

J.M. bon est-ce qu'on met tout de suite Janine and..

• *non, non pas deux affectations l'une derrière l'autre...vous affecterez* clause3 *après... vous affectez d'abord* C31, *et ensuite vous affecterez* clause3, *après... bon alors continuez...*

J.M. bon on va voir...

V. ça fait drôle...

J.M. même au niveau logique... bon, au niveau traduction ?

V. bon si Janine vient, il faut inviter Luc ou Jean

J.M. si j'invite Janine..

V. Janine... Janine and...

J.M. Janine and..

V. ..Jean and not..

J.M. Janine and Jean , clause32 Janine and Luc, clause33 euh Janine and not Jean and not Luc...non..

V. si...

J.M. not Janine and not Jean not Luc...

V. oui ... donc Janine and Jean.. euh ça c'est la clause1...

J.M. à mon avis, tu vas mettre C32...

V. C32 ...euh .. Janine and Luc...

La décomposition du calcul en trois affectations n'est pas comprise.

nous insistons à l'aide d'un exemple pris dans le domaine numérique.

J.M. (vers nous) il y a un problème d'affectation ici (concerne Clause3) ; je ne vois pas ce que vous voulez dire.

• *suppose que tu as un gros calcul à faire... par exemple* = 1 + (3 * (2 + 2 * 4*7)) *bon tu peux l'écrire comme ça... tu peux aussi écrire d'abord* x:=4*7 *ensuite* y:=2*x...

J.M. ah oui, oui, oui...et puis après...

• z:= 2+ y...

J.M. je crois que j'ai compris...

• *ensuite* a:=3*z...

J.M. voilà après la clause3 ça va être le test qui prend...

• *voilà, un calcul qui prendra en compte les trois autres...*

J.M. voilà, mais comment on appelle ça... comment on peut...

• *vous pouvez bien faire un calcul avec, c'est pas un test, c'est un calcul... c'est pas un test, en gros c'est pas une alternative, y a pas un* IF... THEN.. *mais c'est un calcul, c'est-à dire y a des* OR, *des* NOT, *des* AND, *ce que vous voulez...hein...*

J.M. (à **Valérie**) tu as machiné le problème... oui... bon alors, on va faire, tu vas enlever clause31...

V. clause31...

J.M. enfin tu crée une ligne...

V. où çà, euh.. en face de clause3...?

J.M. clause3:=...

V. voilà...

J.M. il faudrait remonter d'une ligne, sans que...

...

V. clause3= si euh c31...and...

Pour **Valérie**, il s'agit de construire une alternative.

J.M. C31 OR !

Jean-Manuel a bien compris qu'il s'agit d'une décomposition.

V. and..?

J.M. non OR !

V. oui, OR, oui, oui maintenant...

J.M. oui, oui, non, non !

V. oui maintenant, c'est OR tu as raison... je mets C31 or

J.M. C32 or C33.. ah, oui mais non, il faut le mettre après... (le calcul de C31, C32, C33)... Il faut le mettre après les trois autres...

(incompréhension de **Valérie** : elle ne réinvestit pas ce qu'elle saurait faire s'il s'agissait d'un calcul sur un type connu).

• (à **Jean-Manuel**) *explique lui, pourquoi il faut le mettre après...*

V. parce qu'il faut les calculer avant...

J.M. c'est clair...

• *oui, si on veut...*

J.M. ...sinon après, il va pas savoir comment...

V. (tape) bon ça va aller ...C31 or C32 ...or C33

J.M. à mon avis, il faut passer une ligne, ce serait plus clair... ensuite on va monter ...déclarer les autres...

...

• *vous sauvez, et vous passez sur imprimante, parce que ça va être l'heure ; si vous avez encore une seconde, je vais vous montrer comment on peut simplifier...*

Le manque de temps laisse subsister deux erreurs dans l'écriture de la CLAUSE3.

J.M. on va peut-être compiler...

(erreur à la compilation)

• *ah oui, vous n'avez pas déclaré* C31, C32

J.M. donc booléens...

V. booléens...

• *vous pouvez les rajouter en début de ligne* (la ligne où sont déclarés les booléens).

J.M. oui, tu rajoutes avant... C31, C32...

V. avant? là haut? avant donc?

J.M. oui là

• *dans la liste des booléens.*

J.M. pourquoi il y a clause 3, clause 4, clause 5 ?

• *parce que j'avais prévu que vous pourriez faire des sous-clauses.*

J.M. oh bon alors c'est bon...

• *on peut les enlever... ....*

J.M. bon alors, c'est bon , on arrête...

7. Protocoles suite à la recherche sur papier du second problème (village)

On trouvera ci-dessous une description des trois protocoles constitués à partir des traces écrites laissées par des élèves..

Protocole C4 (Hervé)

Hervé a imaginé une représentation du schéma du village, sous forme d'un tableau à double entrée : le premier indice est appelé "point de départ", le second "point d'arrivée" ; ainsi, par exemple `TAB[0,1]`, `TAB[0,2]`, `TAB[0,3]` ont respectivement pour valeur 1, 3, 4 car de la maison (n° 0) il est possible d'aller au café (n°1), à l'école (n° 3) et à la mairie (n° 4). De certains bâtiments, il ne part pas trois chemins ; par exemple, de l'Eglise (n°2) on peut aller à la Mairie (n°4) et à l'école (n° 3). C'est pourquoi, `TAB[2,1]` a pour valeur 3, et `TAB[2,2]` a pour valeur 4 ; l'élève affecte à `TAB[2,3]` la valeur 0, ce qui veut dire pour lui qu'il n'y a pas de troisième chemin partant de l'Eglise. Mais ce choix conduit à une ambiguïté : en effet, la valeur 0 pour `TAB[1,3]` signifie qu'il y a un chemin du café (n°1) à la maison (n°0).

L'élève prévoit une "quatrième colonne" désignée par "emprunt", sans rapport avec la représentation du schéma du village ; elle doit permettre de vérifier que le parcours entré par l'utilisateur ne comporte pas deux fois le même bâtiment, ce qui n'est pas demandé par l'énoncé. L'élève initialise cette colonne à 0, dans l'intention de

mettre à 1 l'élément `TAB[N,4]` la première fois que le bâtiment n°N sera rencontré dans le parcours.

Protocole C5 :(Denis)

Comme pour le premier problème (**invitation**), l'algorithme est ici un texte un français, écrit sous forme d'une suite d'actions ; le calcul de la possibilité d'un parcours est évoqué par la mention "on vérifie" sans indication sur la façon dont la machine peut procéder pour cette vérification : "on vérifie (...) que la suite est possible, qu'il existe un passage direct d'un n° à l'autre (...)".

Le trajet "Eglise-Café" est envisagé comme un cas particulier : "on vérifie qu'il ne va pas du 2 au 1", à cause, sans doute, du sens unique.

Le type des diverses données est indiqué: "un tableau de nombres" pour le parcours, des booléens pour chacune des conditions à vérifier, mais la question de la codification du schéma n'est pas évoquée.

Protocole C7 : (Alain)

Ce protocole est un début d'écriture d'un programme en **Pascal** ; il est rayé, ce qui indique que l'élève est arrivé à une difficulté telle qu'il a abandonné sa recherche. Cependant, il témoigne d'une tentative d'écriture d'un programme à l'aide d'alternatives en chaîne : le parcours est représenté par un vecteur d'entiers (`rep[1]` à `rep[4]`) ; l'élève a tenté d'enchaîner jusqu'à trois alternatives, dans une écriture assez confuse, avant d'abandonner.

8. Protocoles suite à la recherche du second problème (village) en travail dirigé avec ordinateurs

8.1 Un programme utilisant une codification du plan du village à l'aide d'un tableau d'entiers (Protocole D4 première version: HERVE

Le tableau est initialisé comme dans l'analyse. Le trajet est représenté par 4 variables numériques `a, b, c, d` . Une procédure contrôle que chacun des 4 trajets élémentaires est possible, et que le premier sommet n'a pas encore été utilisé ; chacun de ces contrôles prend la forme d'une alternative ; dans la branche "alors", le passage au premier sommet est marqué ; dans la branche "sinon", une variable numérique globale "`possible`" reçoit la valeur 0 ou 1. Par exemple, pour le passage de `b` à `c` :

```
if (c=tab[b,1] or c=tab[b,2] or c=tab[b,3] ) and tab[b,4]<>1
then tab[b,4]=1 else possible:=0.
```

La variable "`possible`" étant initialisée à 1, si elle conserve cette valeur à l'issue du contrôle, c'est bien que le trajet est possible. Il s'agit donc de la forme de programmation que nous avons qualifié de "remplacement conditionnel". Ici les alternatives sont complètes, mais la première branche n'est pas constituée d'une affectation à la variable résultat; elle sert à marquer le passage à un point du trajet.

8.2 Trois programmes écrits à la suite d'une incitation à employer une codification du plan du village à l'aide d'un tableau de booléens

Protocole D1 :

```
procedure entreedesdonnees;
 var
```

```
    a,b,c,d:integer;
  begin;
    vrai:=false;
    writeln('Donnez moi le chemin que vous voulez suivre');
    readln(a,b,c,d);
    if a<>d then
   vrai:=(chemin[0,a] and chemin[a,b] and chemin[b,c] and
   chemin[c,d] and chemin[d,0];
  end;
begin
 init;
 entreedesdonnees;
 if vrai then
  writeln('chemin possible');
   else
  writeln('chemin impossible');
end.
```

Protocole D4 seconde version (Hervé R.)

```
writeln('entrez les 4 chiffres');
readln(a,b,c,d);
possible:='possible';
if not chemin[0,a] then possible:= '-------impossible1  ';
if not chemin[a,b] then possible:= '-------impossible2  ';
if not chemin[b,c] then possible:= '-------impossible3  ';
if not chemin[c,d] then possible:= '-------impossible4  ';
writeln(possible);
```

Protocole D7 : (Alain)

```
procedure question;
 var
  i,j:maison;
  vrai:boolean;
 begin
  for i:=1 to 4 do
    begin
     for i:=1 to 4 do
       begin
  vrai:=chemin[i,j];
       end;
    end;
 end;
     if vrai then writeln('chemin possible') else
writeln('chemin impossible');
  end;
begin
 init;
 question;
end.
```