Représentations mentales des données informatiques chez les débutants en programmation

Jean-Baptiste Lagrange

REPRÉSENTATIONS MENTALES DES DONNÉES INFORMATIQUES ET DIFFICULTÉS D'ACQUISITION CHEZ DES DÉBUTANTS EN PROGRAMMATION
Seconde partie : observation d'élèves de l'option informatique

Jean-Baptiste LAGRANGE

Ce texte est la suite de l'article paru dans le dernier numéro de l'EPI ; cet article présentait un cadre d'hypothèses pour l'étude des difficultés de débutants. J'analyse ici, à la lumière de ces hypothèses les résultats d'une observation d'élèves de Seconde et Première Option Informatique.

1. PROBLÈMES POSÉS AUX ÉLÈVES

L'observation sur les chaînes de caractères a porté sur des élèves de Seconde n'ayant pas subi de sélection particulière. L'observation sur les booléens a porté sur des élèves de Première appartenant aux sections scientifiques. J'ai fait passer des épreuves sur papier et des entretiens sous forme de travail dirigé par groupe sur ordinateur, dans l'environnement habituel de la classe, sur les problèmes présentés ci-dessous (figures 4, 5 et 6), ou d'autres peu différents.

1.1. Des problèmes utilisant les chaînes de caractères

Des problèmes simples sur les mots font intervenir la concaténation et la fonction sous-chaîne (problème ONJOURB, figure 4) ; ils sollicitent la compréhension des fonctions sur les chaînes, ainsi que des éléments numériques qui y interviennent et les rapports entre cette structure et les éléments généraux du langage (que nous étudions plus loin).

Dans d'autres problèmes la tâche consiste à rechercher une information dans une chaîne ; la programmation fait intervenir des opérations sur les ordinaux et cardinaux : par exemple, le problème ALPHABET (figure 4) conduit à programmer une "translation" : on passe

du rang d'une majuscule au rang de la minuscule correspondante par "translation" de cardinal 26. Plus généralement, *c* étant un cardinal fixé, la translation de cardinal *c* est l'application qui à un ordinal *n* fait correspondre l'ordinal *n*+*c*. La translation constitue un procédé d'accès à des informations organisées dans une chaîne, ou, plus généralement, dans une mémoire séquentielle.

Figure 4
Des problèmes simples sur les lettres et les mots

problème 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".

problème ALPHABET
la chaîne "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz" étant donnée, écris un programme pour que l'utilisateur, ayant entré au clavier un numéro et répondu à la question "Majuscule ou Minuscule", l'ordinateur affiche la lettre correspondante de l'alphabet.

Des problèmes font intervenir l'itération à un point de sortie : classiquement, on propose des problèmes de comptage d'occurrences, mais comme, dans ces problèmes, le résultat est cardinal, la structure de chaîne n'est pas suffisamment sollicitée, et la nécessité d'une itération résulte directement de la situation. J'ai proposé les problèmes de "*recherche-translation*", comme susceptibles de davantage faire intervenir les représentations au sujet des chaînes. De façon générale on a une chaîne constante où les informations sont groupées par couple. La connaissance du premier élément du couple (information donnée) permet, par une translation d'index d'obtenir l'autre élément (information résultat). En voici un exemple (problème CRYPTAGE figure 5).

Figure 5
Un problème de *recherche translation* (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.

Ces problèmes conduisent à construire "en situation" une première forme de la fonction position à partir d'un jeu réduit de fonctions de base, puis à utiliser la translation. On retrouve donc la problématique énoncée ci-dessus, d'un type de données comme structure abstraite à construire.

1.2. Des problèmes utilisant les booléens

A partir de l'analyse du type booléen présentée en première partie, on peut penser s'inspirer des disciplines technologiques pour construire des problèmes faisant intervenir des objets de type booléen : un dispositif associant des objets à deux états étant donné, on peut demander un programme simulant la réponse du dispositif à des états données de ces objets. Mais ce type de problème présente le désavantage de ne pas correspondre pour tous les élèves à une situation connue. Dans une perspective proche, j'ai fait le choix d'une situation où le type booléen code la présence ou l'absence de personnes, les relations logiques à vérifier étant énoncées sous forme de contraintes sociales (problème INVITATION figure 6).

Figure 6
Un problème sur des objets à deux valeurs (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 vérifiée).

2. RÉSULTATS OBSERVÉS

J'ai recueilli à partir de cette observation un nombre important d'erreurs très diverses faites par ces élèves ; LAGRANGE (1991) présente un relevé exhaustif de ces erreurs et les circonstances dans lesquelles elles ont été recueillies. A partir de ce relevé d'erreurs et des hypothèses énoncées ci-dessus, il est possible d'établir une classification des difficultés à l'origine de ces erreurs. Chacune des difficultés est illustrée ci-dessous par un exemple d'erreur, ainsi qu'une indication concernant la population sur laquelle elle a été observée [1].

2.1. Les plans des objets

2.1.1. L'utilisation du langage (affectation, expressions) sur les différents objets

Certaines difficultés rencontrées sont relatives aux rapports entre les objets et les éléments généraux du langage.

. Les expressions sur les chaînes peuvent être comprises non comme des objets résultat d'un calcul, mais comme le codage d'actions sur des objets internes au dispositif. Le rôle de l'affectation dans les changements d'état du dispositif est ainsi ignoré.

Une expression se trouve ainsi comme instruction isolée, ou, plus subtilement dans une affectation ou une instruction de sortie qui, à l'analyse, se révèlent n'avoir pas de signification pour l'élève :

> Ainsi, on trouve un appel de la fonction sous-chaîne de façon isolée dans une ligne de programme, ou dans une affectation *A<-sous-chaîne(.....)*,

1 Le langage de programmation utilisé par les élèves est BASIC ou PASCAL, suivant les classes. Dans la suite, les exemples d'écritures sont transcrits dans un pseudo-langage faisant intervenir les fonctions chaînes, les opérateurs booléens, l'affectation...

l'identificateur *A* n'étant pas réutilisé dans la suite du programme : pour l'élève, cette écriture ne représente pas une valeur destinée à être utilisée dans un calcul ultérieur, mais a pour fonction d'"extraire" physiquement une partie de la chaîne [2].

Ce comportement est observé chez 4 élèves sur 10 dans une épreuve comportant le problème ONJOURB (figure 4) à l'issue d'une trentaine d'heures d'enseignement.

Cette difficulté se manifeste particulièrement lorsque les expressions sont composées sur plusieurs niveaux ; dans ce cas, le rôle d'action peut également être attribué à une sous-expression, c'est-à-dire à une expression argument d'une fonction.

Par exemple, de nombreux élèves calculent le dernier caractère d'une chaîne C de longueur L par sous-chaîne (C,L-1,1), au lieu de sous-chaîne (C,L,1). L'argument L-1 dans cet appel de la fonction sous-chaîne code l'action "d'ôter le dernier caractère".

Ce comportement est observé chez 17 élèves sur 52 (1/3) dans une épreuve passée à l'issue de 75 heures d'enseignement.

. Les expressions à valeur booléenne ne sont pas comprises comme des objets résultats d'un calcul : les élèves leur donnent un sens comme premier argument d'une alternative (c'est-à-dire comme conditions), mais éprouvent des difficultés à donner une signification à l'affectation et aux calculs sur ces expressions.

Ayant proposé le problème INVITATION (figure 6) à une classe de seconde année (9 élèves) après un enseignement sur les booléens, je n'ai obtenu dans aucun protocole spontané le calcul d'un résultat par affectation d'un booléen.

Un autre exemple courant est la réticence à utiliser l'affectation directe d'un booléen par booleen :=<condition>, le sujet préférant généralement l'affectation par une alternative : if <condition> then booleen :=true else booleen :=false.

2.1.2. Les représentations des types chaîne et booléen

De fausses représentations des objets dans leur cohérence interne peuvent aussi se manifester, attestées par la difficulté à donner un sens

2 On retrouve ici la singularité du type numérique dont nous parlions plus haut : en effet, dans ce type, les expressions sont en fait les expressions algébriques installées par l'enseignement des mathématiques, et les élèves ne leur attribuent pas généralement le rôle d'action.

aux fonctions des différents types et par des confusions portant sur les objets eux-mêmes.

. De nombreuses écritures sur les chaînes ne respectent pas les régles du type : en fait les arguments des fonctions sur les chaînes sont souvent compris comme des "informations" au statut flou, "données" à l'ordinateur dans le but d'obtenir le résultat souhaité.

> Dans le problème ALPHABET (figure 4), un groupe d'élèves observés calcule la ième lettre minuscule comme sous-chaîne(alphabet,i, minuscule), au lieu de sous-chaîne (alphabet,i+26, 1). Pour eux, les éléments pertinents sont le rang i de la lettre dans l'alphabet, et sa propriété de minuscule ; il n'ont pas conscience de ce que l'appel de la fonction sous-chaîne provoque un calcul sur la chaîne alphabet, calcul pour lequel les arguments doivent répondre à des régles formelles.
>
> Dans une épreuve comportant un problème peu différent de ONJOURB, à l'issue de 75 heures de programmation, les réponses de 33 élèves sur 52 (2/3) s'interprètent comme résultant d'une incompréhension de la signification des arguments des fonctions sur les chaînes.

. Les débutants éprouvent des difficultés à donner une signification aux variables ordinales et à les coordonner avec les cardinaux. Ainsi, les ordinaux sont peu distingués des caractères qu'ils pointent dans les chaînes.

> Par exemple, dans le problème CRYPTAGE (figure 5), Code étant la chaîne constante, on rencontre comme calcul du caractère résultat du cryptage d'un caractère Lettre, l'expression sous-chaîne (Code, Lettre + 1,1) ou même plus simplement Lettre +1. (6 élèves sur 14 dans une épreuve après une cinquantaine d'heures d'enseignement)

Le résultat de la fonction longueur doit être considéré dans certains cas comme cardinal (nombre de caractères de la chaîne) et dans d'autres comme ordinal (position du dernier caractère de la chaîne). Cette dualité conduit à ce que ce résultat est assimilé à l'objet chaîne dont il est la mesure, mais aussi à "la fin de la chaîne" dont il est la position.

> J'ai rapporté ci-dessus l'exemple de sujets qui donnent L-1 pour la position du dernier caractère d'une chaîne de longueur L . Si l'expression L-1 code l'action "d'ôter le dernier caractère de la chaîne", c'est bien que L n'est pas un simple nombre mais bien à la fois "la chaîne toute entière" dont on ôte un caractère et "la fin de la chaîne" puisque le caractère ôté est le dernier.

. Dans la construction d'expressions booléennes, certains connecteurs sont employés plus volontiers que d'autres. Les sujets observés utilisent préférentiellement les connecteurs *AND* et *"différent de"* pour former les parties "conditions" des alternatives ; les connecteurs *OR* (disjonction) et

NOT (négation) sont peu employés. Le connecteur *AND* représente la conjonction logique, mais, intervenant pour former une "condition" , il peut être compris comme proche du "et" du langage naturel. L'opérateur *"différent de"*, appliqué à des booléens, représente la négation de l'équivalence logique, ou le connecteur *XOR* (disjonction exclusive), mais ici, les sujets l'emploient spontanément pour traduire que deux variables ont des valeurs distinctes, sans qu'ils aient nécessairement conscience du caractère logique de ces valeurs : en effet, les élèves ont utilisé cet opérateur sur d'autres types et intégré les expressions formées avec cet opérateur à partir de leur expérience antérieure (mathématiques, langage habituel..).

> Ainsi, dans les productions spontanées pour la résolution du problème INVITATION (figure 6), 7 élèves sur 9 résolvent au moins partiellement la programmation des clauses : l'un d'entre eux n'utilise aucun connecteur logique, les 6 autres utilisent le connecteur AND, et parmi ces 6 élèves, 4 emploient le connecteur différent de (la 2ème clause demandée s'écrit ainsi : SI Marie "différent de" Marc). Dans la suite de la résolution où je les guide vers une utilisation du type booléen, les sujets utilisent le connecteur NOT, mais ne simplifient pas des expressions telles que NOT (A AND B) et même NOT (A différent de B). Seuls deux élèves utilisent le connecteur OR ; leur résolution est l'expression disjonctive des situations élémentaires où la clause est réalisée : dans cette forme, le connecteur OR peut être considéré aussi bien comme exclusif que comme inclusif.

2.2. La notion de type

. En première année, l'enseignement reçu au cours des premières séances, même s'il a porté sur des types différents (nombres, chaînes), n'a pas "installé" les contraintes de type. L'utilisation d'opérations non cohérentes avec le type en est un témoignage.

> Devant isoler le chiffre des unités d'un nombre donné et le comparer à zéro, deux élèves observés en entretien après 40 heures d'enseignement, utilisent, dans leurs productions spontanées, des fonctions chaînes sur ce nombre, et établissent une comparaison entre un caractère et un nombre.

. Les difficultés à donner un sens aux expressions booléennes, et à programmer les entrées-sorties de valeurs booléennes peuvent conduire à l'abandon du type booléen. Dans les représentations spontanées les booléens apparaissent en effet adaptés à la codification des objets à deux valeurs, mais, au moment de l'écriture effective du programme, les sujets abandonnent ce type, au profit d'un type avec lequel ils sont plus familiers.

Les sujets résolvant le problème INVITATION choisissent au départ le type booléen pour codifier les personnes en jeu dans le problème. Les productions à l'issue de la séance font apparaître dans 3 cas sur 9 le maintien du type booléen : en fait, dans ces protocoles, la résolution est à peine amorcée. Les 6 autres élèves résolvent partiellement le problème en codifiant les données à l'aide d'un type numérique (1, si personne présente, 0 si personne absente)

2.3. L'interaction entre objets et traitements

2.3.1. Traitements et objets chaînes

Voici deux schémas (erronés) de résolution d'un problème de "recherche translation" (problème CRYPTAGE figure 5)

- le schéma *"Accès Direct"* suppose possible d'accéder à un caractère de la chaîne par la seule connaissance de la *valeur* de ce caractère, sans recherche itérative, en confondant, en fait, le caractère et sa position dans la chaîne. Il s'exprime par exemple dans le programme suivant :

Résultat <- SousChaine(Code, Donnée+1,1)

Ce schéma comprend donc "un plan du traitement" incompatible avec les contraintes de recherche séquentielle propre au traitement informatique sur une chaîne de caractères. Il comprend également une conception (un "plan") des objets déjà décrit ci-dessus, où objets et ordinaux sont confondus, qui rend possible à la fois l'accès direct au caractère recherché, et le calcul du caractère suivant par addition du nombre 1.

- le schéma *"Iteration Naïve"* se caractérise par l'intégration d'un parcours itératif, sans qu'il y ait compréhension du mécanisme d'évolution des variables dans l'itération. Voici un exemple de programme correspondant à ce schéma relevé dans des productions d'élèves :

Pour I<-1 jusqu'à Longueur(Code)
Sous-chaine <- (Code,Donnée,I+1)
Afficher Donnée+1

La première ligne de ce programme est syntaxiquement un avertisseur de début d'itération, mais la suite montre qu'elle annonce simplement l'intention de procéder à un parcours de la chaîne Code sans s'insérer dans une construction itérative conforme aux contraintes de la programmation.

Dans la seconde ligne, l'identificateur de la fonction appelée est à gauche du signe d'affectation, ce signe étant lui-même suivi des arguments de la fonction ; cette écriture a pour fonction d'isoler le caractère *Donnée* dans la chaîne *Code* (par appel de la fonction *sous-chaîne*), puis d'incrémenter l'index (par la sous-expression *I+1*). Il s'agit bien d'une manifestation de la conception des écritures sur les chaînes repérée ci-dessus, où les expressions sur les chaînes sont comprises comme **codage d'actions.** L'identificateur de variable "résultat de l'extraction", qui serait nécessaire d'un point de vue syntaxique, n'est pas présent dans cette écriture ; en effet, il n'aurait pas de signification dans ce codage d'actions.

La dernière ligne se comprend comme l'affichage du caractère suivant le caractère repéré à la ligne précédente. Cet affichage est rendu possible par la suite d'actions précédente : il suppose en effet qu'il y ait eu auparavant un balayage itératif de la chaîne Code, et l'incrémentation de l'index. On a donc bien un *plan du traitement* intégrant la nécessité d'un parcours itératif, mais ce plan ne prend pas en compte de façon suffisamment précise la mise en oeuvre des capacités de contrôle du dispositif.

> Ayant proposé le problème CRYPTAGE (énoncé figure 5) à une classe de 15 élèves après 50 heures d'enseignement, j'ai obtenu 8 réponses dont 2 correctes. Les 6 autres réponses se répartissent entre les schémas "Accès Direct" (4 élèves) et "Iteration Naïve" (2 élèves) décrits ci-dessus.
>
> Dans une épreuve portant sur 52 sujets après 75 heures d'enseignement, j'ai demandé de compléter certains arguments des fonctions dans un problème de recherche translation : 18 élèves (1/3) donnent une réponse correcte, 19 élèves (1/3) donnent une réponse compatible avec le schéma "Accès Direct", 12 élèves (1/4) donnent une réponse où le parcours itératif est pris en compte, mais sans compréhension de l'évolution du compteur de boucle (schéma "Iteration Naïve")

2.3.1. Traitements et objets booléens

Cette conjonction d'un plan du traitement et de conceptions des objets s'observe également dans la résolution d'un problème portant sur des objets à deux états (problème INVITATION, figure 6)

On peut distinguer trois schémas de solution, que j'explicite partiellement dans le cas de la 3ème clause ("Si j'invite Janine, il faut que j'invite Luc ou Jean, mais on ne peut pas les inviter tous les trois ensemble")

- dans le schéma *"Action Ramifiée"*, les objets sont d'un type familier à l'élève (par exemple numérique), il n'y a pas utilisation des connecteurs logiques, et le résultat n'est pas calculé : il y a simplement une action d'affichage conditionnel qui s'exprime par des alternatives complètes (SI...ALORS..SINON...) fortement imbriquées.

```
SI JANINE=1 ALORS
        SI LUC=0 ALORS
           SI JEAN = 0 ALORS
                       AFFICHER "Pas possible"
                   SINON
                       AFFICHER "Possible"
        SINON...
```

- dans le schéma *"Remplacement Conditionnel"*, le résultat est une variable qui reçoit par affectation initiale une valeur affirmative. Le programme est une suite d'alternatives incomplètes (SI...ALORS..) non imbriquées ; les objets ne sont pas nécessairement booléens, mais, dans la partie condition, le sujet peut employer des connecteurs logiques (essentiellement la conjonction)

```
Résultat <- 1
SI JANINE=1 ET LUC=0 ET JEAN=0 ALORS Résultat <- 0
.....
```

- dans le schéma *"Booléen"*, le programme est un calcul sur des objets booléens sans utilisation de l'alternative

```
Résultat <- NON(JANINE) OU (LUC différent de JEAN)
```

Dans les productions spontanées pour la résolution du problème INVITATION (figure 6), parmi les 7 solutions partielles des clauses, aucune ne relève du schéma booléen, 3 relèvent du schéma "Remplacement Conditionnel", les 4 autres étant des schémas mixtes où sont présents à la fois des Remplacements Conditionnels et des "Actions Ramifiées". Par la suite, j'oriente les élèves vers le schéma "Booléen" en résolvant pour eux la question des entrées/sorties : ils se heurtent alors aux difficultés rapportées ci-dessus (sens de l'affectation de variables booléennes, usage des connecteurs logiques...)

3. INTERPRÉTATION

3.1. Le langage, quand il porte sur des types peu familiers, est utilisé de façon "naïve"

Les difficultés rencontrées dans l'utilisation des éléments généraux du langage sur les objets chaînes et booléens révèlent en effet des représentations du langage où les expressions ne sont pas comprises comme des valeurs résultat d'un calcul : dans ces représentations, les expressions chaînes codent des actions sur leurs arguments ; les expressions booléennes sont des "conditions", c'est-à-dire des formes inspirées des propositions circonstancielles du langage habituel. Ainsi, le programme n'est pas un calcul sur des objets chaînes ou booléens, mais une description métaphorique de ce calcul en terme d'actions, éventuellement conditionnelles. Les élèves qui rencontrent ces difficultés n'ont donc pas pris conscience des propriétés du langage de programmation lorsqu'il porte sur ces types, et utilisent en fait les éléments de ce langage pour coder des formes inspirées du langage habituel.

Dans le cadre impératif où les observations ont été faites, la compréhension de l'affectation est en quelque sorte le pivot de cet ensemble de représentations. Face à une tâche impliquant l'appel successif de fonctions sur les chaînes (par exemple ONJOURB, figure 4), on rencontre des sujets peu nombreux qui maîtrisent très vite la composition des fonctions et évitent ainsi le recours à l'affectation (1 à 2 sur 15, en moyenne, dans les classes observées). Parmi les autres élèves, ceux qui produisent des expressions complexes erronés codant, pour eux, une suite d'actions sur le dispositif, sont aussi ceux qui utilisent le moins l'affectation. Dans le cas de ces élèves, l'affectation, quand elle est présente, sert seulement pour la correction syntaxique [3]. Au contraire, les élèves qui utilisent volontiers l'affectation pour décomposer les calculs semblent avoir une meilleure compréhension du mécanisme de transfert de valeur en jeu dans l'affectation, et mieux intégrer la signification des expressions en tant que valeur.

Les difficultés repérées ici tiennent bien aux caractéristiques générales des dispositifs informatiques et des langages par lesquels on

3 On peut penser que, pour ces élèves, les expressions se suffisent à elles-mêmes. Mais leur attention a été attirée, soit par le professeur, soit par les erreurs de compilation, sur la nécessité d'inclure ces expressions dans une affectation ou une instruction de sortie. En conséquence, on peut trouver des affectations ou instructions de sortie syntaxiquement correctes, mais sans signification pour l'élève (par exemple, l'identificateur de variable à gauche du signe d'affectation n'est pas réutilisé dans la suite).

les programme. Elles ne sont pas introduites de façon artificielle par l'utilisation de langages impératifs faisant appel à l'affectation, et on peut penser qu'elles apparaîtraient sous d'autres formes dans d'autres types de langages. J'ai observé par exemple (Lagrange, 1986) des étudiants programmant en LOGO incapables de distinguer dans quelles situations il convient qu'une procédure se termine par ECRIS <résultat> (et ait par conséquent le statut de procédure) plutôt que par l'instruction RENDS <résultat> (qui lui donnerait le statut de fonction).

3.2. Les représentations des objets chaînes et booléens sont également marquées par le contexte "naïf"

En effet, les écritures erronées sur les chaînes révèlent des représentations où le traitement des informations par le dispositif n'est pas un calcul, et où par conséquent, l'insertion de ces informations dans les écritures n'a pas à obéir à des règles précises : pour le sujet qui produit ces écritures, l'essentiel est que le dispositif ait tous les éléments nécessaires pour "comprendre" la requête. Cette conception "non calculatoire" du traitement des informations autorise, comme dans le contexte familier, les confusions entre objet et mesure, entre objet et position.

De même, les opérateurs booléens les plus volontiers employés sont les connecteurs différence (différent de), et conjonction (AND), dont l'utilisation peut s'appuyer sur des représentations familières. Les sujets hésitent au contraire dans l'emploi de la disjonction (OR) vraisemblablement à cause de l'ambiguïté exclusif-non exclusif qui est ici critique à la différence du contexte familier. De même, ils n'utilisent pas la négation (NOT) qui n'existe pas en tant que telle dans le contexte familier, et qu'il est possible d'éviter en programmation lorsqu'il s'agit de construire des alternatives : il est souvent possible de nier une condition sans employer la négation [4] ou de "retourner l'alternative" [5].

3.3. Le typage des objets n'est pas intégré dans les représentations

Un sujet expérimenté établit son schéma de résolution en envisageant de façon cohérente le type des données qu'il projette d'utiliser pour la codification, et les opérations qu'il se propose d'effectuer

4 *A=B* est nié en *A différent de B*

5 *SI NOT <condition> ALORS <action1> SINON <action2> se simplifie en SI <condition> ALORS <action2> SINON <action1>*

sur ces données. Au contraire, les sujets débutants observés appliquent à des entités numériques de façon indifférenciée des fonctions chaînes et des opérateurs numériques. D'autres prévoient le type booléen pour codifier les objets d'un problème, mais ne planifient pas dans le même temps un traitement adapté à ce type.

Ici aussi, on peut s'interroger sur ce "polymorphisme" que les sujets débutants supposent aux objets informatiques : est-il ou non compatible avec un S.R.T. informatique ? Les concepteurs de LOGO ont par exemple tenté d'adapter le langage à cette conception spontanée (les entiers en LOGO sont en fait la chaîne de leurs chiffres, les valeurs booléennes sont des chaînes), analysant implicitement la contrainte du typage comme artificielle. On peut au contraire considérer l'intégration du typage des objets d'une part comme un point d'appui pour de meilleures représentations du dispositif, et d'autre part comme une aide à la conception : pour résoudre par exemple le problème INVITATION, un sujet ayant une représentation efficace du type booléen pourra se laisser guider par le type des données et du résultat.

3.4. Résistance des représentations

Comme je l'ai constaté au cours d'entretiens, les difficultés rapportées ci-dessus résistent aux explications et rappels à la syntaxe de l'enseignant, au diagnostic d'erreur renvoyé par l'ordinateur (erreurs de compilation, résultat d'exécution non conforme à l'anticipation...) Deux facteurs semblent pouvoir expliquer cette résistance :

¨ De nombreuses écritures peuvent résulter de plusieurs des représentations erronées décrites ci-dessus. Par exemple, la position *L-1* pour le calcul du dernier caractère d'une chaîne de longueur *L* se justifie à la fois par une difficulté à articuler le double statut ordinal/cardinal de la fonction longueur, et aussi par la compréhension de l'écriture *L-1* comme codage de l'action "enlever un caractère". Les représentations sont donc d'autant plus résistantes qu'elles se renforcent mutuellement pour une même écriture.

¨ Dans de nombreux problèmes ou exemples, les réponses exactes peuvent être compatibles avec les représentations erronées repérées ci-dessus. Par exemple, une écriture correcte des arguments des fonctions chaîne, donne bien tous les éléments nécessaires à la détermination du résultat, et n'entre donc pas en contradiction avec une conception naïve de ces fonctions. En fait, les représentations erronées apparaissent seulement dans des exercices où la structure de chaîne est fortement

sollicitée, avec des réponses se différenciant nettement des conceptions intuitives.

3.5. Objets et traitements

Dans les problèmes mettant en jeu l'itération et les chaînes de caractères (problème CRYPTAGE figure 5), les schémas de solution erronés conçus par les sujets observés s'analysent comme la conjonction d'un plan des objets et d'un plan du traitement.

Dans le schéma *"Accès Direct"* le plan du traitement ne comporte pas d'itération ; le sujet pense possible d'accéder directement à la position du caractère donné dans la chaîne de cryptage. Le plan des objets est quant à lui marqué par la conception non-calculatoire des objets chaîne, vue ci-dessus, qui permet qu'un caractère soit assimilé à sa position dans une chaîne. Ces plans, erronés dans le contexte de la programmation, sont cependant cohérents entre eux. D'autre part, si l'on se place dans le contexte familier, la chaîne de cryptage étant suffisamment courte, un traitement peut consister à repérer par appréhension globale le caractère donné dans la chaîne de cryptage, puis à "passer au suivant". Le plan du traitement, et le plan des objets du schéma *"Accès Direct"*, sont donc tout à fait compatibles avec ce contexte familier.

Le schéma *"Iteration Naïve"* traduit l'intention d'un parcours de la chaîne de cryptage sans constituer le codage effectif d'un mécanisme itératif et, dans ce schéma, les expressions et sous-expressions chaînes codent des actions. A la différence du schéma *"Accès Direct"*, le plan du traitement satisfait la contrainte du parcours de la chaîne, mais l'expression de ce traitement témoigne d'une prise de conscience insuffisante des contraintes concernant le contrôle de l'exécution par le dispositif, et le plan des objets est, quant à lui, marqué par les difficultés analysées ci-dessus, à comprendre et à exprimer le calcul des objets par le dispositif. Le sujet utilise en fait les éléments du langage de programmation pour tenter d'exprimer un traitement et des manipulations d'objets qui sont trop éloignés d'un traitement informatique. On a donc ici une conception "naïve" du langage qui s'exprime de façon cohérente sur le traitement et sur les objets.

On retrouve cette interaction entre traitements et objets dans les schémas de solution au problème INVITATION. Dans le schéma *"Action Ramifiée"*, le sujet considère le traitement comme une arborescence complexe d'actions ; ce plan du traitement ne nécessite aucunement la prise en compte d'objets : conditions et actions sont des formes proches

du langage habituel. Dans le schéma *"Remplacement Conditionnel"*, les actions sont ordonnées séquentiellement ; ce "plan" suppose de considérer le résultat comme un objet, et sollicite davantage la construction des conditions. Dans le schéma *"Booléen"*, les objets du problèmes sont des objets booléens clairement envisagés comme calculables ; le traitement se réduit au calcul. Chacun des schémas associe donc de façon étroite un *"plan du traitement"* et un *"plan des objets"*.

De façon générale, les schémas les plus rencontrés (*"Accès Direct"* et *"Action Ramifiée"*) s'inspirent de traitements dans le contexte habituel : ces schémas sont le résultat de l'évocation conjointe d'un plan de traitement et d'une conception des objets adaptés à la résolution dans les "situations connues".

Par ailleurs, l'expression d'un traitement sur des objets d'un type donné apparaît très entachée des difficultés propres à ce type, elles-mêmes déterminées par les représentations mentales des objets de ce type. Les difficultés rencontrées dans le transfert de capacités algorithmiques d'un domaine d'objets à l'autre (par exemple du domaine graphique au domaine des listes numériques dans (Samurcay et Rouchier, 1990), pourraient ainsi s'interpréter par l'intervention des représentations propres à ce domaine.

4. UNE EXPÉRIMENTATION PÉDAGOGIQUE EN SECONDE

A partir de la mise en évidence des difficultés ci-dessus, j'ai tenté de préciser quelles directions d'expérimentation pédagogique on peut mettre en œuvre pour les réduire. La situation pédagogique de résolution d'un problème de programmation par des débutants en situation scolaire a été peu analysée jusqu'à présent. C'est pourquoi j'ai préféré, plutôt que d'innover dans ce domaine, centrer l'intervention didactique sur deux aspects précis de cette situation : la conception des problèmes, et l'intervention pédagogique de l'enseignant dans une situation de résolution par groupe en présence de l'ordinateur.

4.1. Les problèmes

J'ai fait ci-dessus l'hypothèse que l'acquisition d'un S.R.T. adapté à la programmation passe par un travail sur les difficultés résultant des représentations naïves. Les problèmes doivent être construits pour faire rencontrer aux élèves ces difficultés, et organisés en progressions de façon à permettre cette construction progressive d'un S.R.T. J'ai donc

écrit des séries d'énoncés, la tâche de l'élève pour chacun de ces énoncés étant précisément repérée et mise en relation avec les difficultés analysées ci-dessus. Comme les problèmes utilisés pour l'observation, ces énoncés s'appuient sur des situations ayant un sens intuitif pour l'élève, à partir desquelles il peut évoquer des plans par analogie, mais où le raffinement de ces plans conduit à leur remise en cause ou leur adaptation visant à les rendre compatibles avec le dispositif.

4.2. La situation pédagogique de résolution

Le programme de l'option informatique (MEN, 1987) privilégie l'analyse du problème comme préalable à la programmation. Or, s'il semble effectivement difficile que le sujet conçoive une solution sans avoir "analysé l'exercice", l'explicitation d'une telle analyse paraît problématique dans le contexte des exercices proposés au débutant : dans certaines classes, il s'agit d'exprimer en langage naturel un procédé de résolution "manuel", c'est-à-dire au niveau des situations connues, ce qui est, comme nous l'avons vu plus haut, de peu d'intérêt vu le caractère immédiat de ce traitement. Il peut s'agir aussi d'évoluer vers l'explicitation d'un processus plus "informatique", mais l'élève qui perçoit un schéma de solution présentant un minimum de compatibilité avec le dispositif, préfèrera l'exprimer dans le langage de programmation ou dans un langage proche, ce qui fait que l'analyse sera en fait un programme ou une ébauche de programme, et, pour ma part, je m'en suis tenu à cette exigence.

D'autre part, le programme de l'option insiste sur les phases collectives où en dehors de l'ordinateur : il semble bien que, dans ce cadre, ce soit au professeur d'instituer, dans son seul discours, les spécificités du traitement informatique par rapport aux situations connues. Le débutant peut ainsi éprouver des difficultés à comprendre la nature exacte de ces spécificités et à les distinguer d'exigences didactiques.

Je pense au contraire important que les débutants soient confrontés au dispositif ordinateur+langage, en programmant dans la perspective d'un programme utilisable et non en fonction d'exigences didactiques comprises par eux comme extérieures à l'activité. Dans cette phase de travail sur ordinateur, le professeur intervient de façon individualisée à partir de difficultés rencontrées par les élèves. J'ai montré ci-dessus que ces difficultés ne résultent pas de simples oublis de la syntaxe mais bien de S.R.T. non encore adaptés à la programmation. Pourtant les interventions portant sur la syntaxe sont les plus faciles

pour l'enseignant car elles ne demandent pas d'analyse de la réponse de l'élève ; j'ai de bonnes raisons de penser que ces interventions peuvent apporter une correction de la syntaxe, sans intervenir sur les S.R.T., et ainsi conduire à renforcer des S.R.T. naïfs en leur faisant correspondre des expressions compatibles avec la syntaxe du langage [6]. L'intervention de l'enseignant auprès des élèves peut être davantage pertinente si elle se centre sur la sémantique des écritures produites par les élèves [7]. Cette intervention sur la sémantique suppose de la part de l'enseignant une capacité d'analyse des productions des élèves, reposant sur la connaissance des S.R.T. ou au moins sur des hypothèses à ce sujet.

4.3. Résultats

A partir des principes ci-dessus, j'ai mené une première expérimentation limitée à une dizaine de séances s'insérant dans le déroulement normal du second trimestre de la classe de Seconde Option Informatique, en observant plus particulièrement deux groupes de deux élèves présentant de grandes difficultés. Voici les résultats de cette expérimentation.

4.3.1. Le typage des objets

On a vu ci-dessus que les débutants choisissent le type en considérant la "nature" des données, plus que le jeu de fonctions qu'ils prévoient d'utiliser. A l'issue d'un travail à partir d'exercices remettant en cause ce choix spontané, les élèves observés, pourtant très en difficulté au départ, résolvent un problème imposant le choix successif du type chaîne puis du type numérique pour le codage d'un même nombre. Ils progressent donc notablement dans ce domaine, et j'ai observé que ce progrès a une influence positive sur leurs représentations : la nécessité de respecter le type dans les expressions, la distinction entre caractère et index, sont intégrés en fonction d'une meilleure compréhension des objets et non comme des nécessités syntaxiques.

4.3.2. L'affectation dans le contexte du traitement des chaînes

J'ai indiqué ci-dessus que l'utilisation de l'affectation pour la décomposition des calculs permet à l'élève à la fois de mieux maîtriser l'affectation et d'éviter des expressions trop complexes auxquelles il lui est difficile de donner une signification, autre que naïve. Dans

6 Pour un exemple de ce processus, voir la note 7 ci-dessous.

7 Ce qu'on peut traduire par : travailler la question *"Que veut on faire par cette expression ?"*, plutôt que *"Cette expression est-elle correcte ?"*

l'expérimentation, j'ai donc orienté les élèves vers ce type de décomposition ; ces élèves font de nombreuses erreurs au départ, mais, à l'issue de cette expérimentation, ils manifestent une certaine maîtrise des expressions sur les chaînes et donnent un sens à l'affectation.

4.3.3. Les représentations des objets chaîne et l'acquisition de l'itération

A la suite des résultats ci-dessus, j'avais pensé intéressant, dans le cadre du traitement itératif des chaînes, de jouer sur l'interdépendance entre conception des objets et plan du traitement pour créer un déséquilibre : en effet, on peut s'appuyer sur une adaptation dans le domaine des objets pour conduire les élèves à remettre en cause leur plan du traitement. Bien qu'un certain progrès se manifeste aussi dans le domaine de l'itération, les élèves ont rencontré de nombreuses difficultés au cours de l'expérimentation et ne les ont pas toutes résolues. Il faut en fait considérer qu'il existe une variété importante de représentations spontanées des objets chez les débutants ; leur mise en évidence par le professeur, et le nécessaire travail de remise en cause et d'adaptation par le sujet lui-même nécessitent certainement un temps plus long que la dizaine de séances consacrée à l'expérimentation.

5. CONCLUSION

L'option informatique des lycées a constitué un terrain fort intéressant d'observation et d'expérimentation. Ce texte s'attache à contribuer à un cadre d'analyse des difficultés rencontrées. Il montre que ces difficultés sont clairement du domaine du passage à l'abstraction. Loin de condamner l'idée d'un enseignement de l'informatique à des débutants "tout public", il met en évidence sa nécessité et son caractère formateur. De plus, il semble bien que des difficultés de même nature sont rencontrées lors de l'utilisation d'outils informatiques pour l'enseignement d'autres disciplines [8] : les représentations du dispositif sous-jacent à l'outil informatique, en particulier celles portant sur les objets manipulés par ce dispositif, interfèrent alors fortement avec l'activité propre à la discipline, et créent d'autant plus de difficultés que la modification de ces représentations ne fait pas partie des objectifs visés par l'enseignement. Le travail présenté dans cet article se situe dans une perspective permettant de nombreux prolongements : étude des

8 Voir par exemple les difficultés rencontrées lors de l'utilisation d'outils de tracé et de description de figures dans l'enseignement de la géométrie (ARTIGUES, 1991)

représentations dans divers contextes (types d'objets, langages, "paradigmes" de programmation, outils...), étude plus approfondie de la situation pédagogique de résolution, production et expérimentation de progressions à plus grande échelle... Les premiers apprentissages en informatique constituent donc un domaine de recherche largement ouvert.

Jean-Baptiste LAGRANGE

BIBLIOGRAPHIE

ARTIGUES M. (1991) « Analyse de processus d'enseignement en environnement informatique », *Actes de l'Université d'été Informatique et Enseignement de la Géométrie* IREM de TOULOUSE.

BARON G.L.(1988) *L'informatique comme discipline scolaire.* PUF.

DUCHATEAU CH. (1991) *Images pour programmer* De Boeck Université.

HOC J.M. (1987) *Psychologie cognitive de la planification* Editions PUG.

LAGRANGE J.B. (1991) *Des situations connues aux traitements sur des données codifiées...* Thèse de Doctorat Université PARIS VII.

MEN (1987) *Option Informatique Classes de seconde, première et terminale*. CNDP.

SAMURCAY R. (1985) « Signification et fonctionnement du concept de variable informatique chez des élèves débutants », *Educational Studies in Mathematics* n°16 Kluwer Academic Publishers.

SAMURCAY R. ROUCHIER A. (1990) « Apprentissage de l'écriture et de l'interprétation des procédures récursives », *Recherches en didactique des Mathématiques*, Vol 10 2.3. Editions la Pensée Sauvage.