RENAN SAMURÇAY

SIGNIFICATION ET FONCTIONNEMENT DU CONCEPT DE VARIABLE INFORMATIQUE CHEZ DES ELEVES DEBUTANTS[1]

RESUME. Dans le cadre d'une problématique d'ensemble sur les difficultés cognitive posées par l'acquisition des concepts informatiques, nous avons étudié chez des élèves débutants, la résolution de problèmes relatifs à la représentation et au traitement des variables informatiques, dans une tâche de complétion de lignes manquantes dans un programme. Les résultats permettent de discuter les relations entre d'une part la nature des solutions apportées par les élèves, d'autre part les propriétés du champs conceptuel en jeu et la maniére dont les concepts ont été introduits.

ABSTRACT. Our general purpose is to study the cognitive problems encountered by subjects in their acquisition of programming concepts. In this paper we focus our attention on the concept of variable and its use in problem-solving by naive students (15–16 years old) who had previously followed a didactical experience. We offered to the students a set of computer program texts in which one or more instructions were missing: they were asked to complete these instructions. The results are discussed in terms of the nature of the solution given by the student, the properties of the conceptual field and the way in which these concepts were introduced.

1. INTRODUCTION

Bien que la psychologie cognitive s'intéresse beaucoup actuellement à l'activité de programmation informatique, elle s'interroge rarement sur les processus d'acquisitions des concepts et procédures informatiques dans leur spécificité. Une grande partie des travaux diffusés jusqu'ici (Anderson *et al.*, 1984; Schneiderman, 1980; Jeffries, 1982) concernent essentiellement l'étude des stratégies générales de programmation sans que soient pris en compte les concepts que celles-ci font intervenir.

Par ailleurs, les nombreuses expéreinces d'introduction "précoce" de la programmation dans l'enseignement sont présentées souvent du point de vue de ce que fait ou peut faire l'enseignant, elles apportent peu d'informations sur l'acquisition que font les élèves des contenus d'enseignement spécifiques. De plus ces travaux portent majoritairement sur l'utilisation du langage LOGO-graphique qui différe sensiblement par sa structure, des langages "impérarifs" comme BASIC, PASCAL, etc. . .

Il existe à l'heure actuelle très peu d'études sur les phénomènes qui interviennent pendant la période d'"alphabétisation"[2] concernant l'acquisition des concepts de la programmation. Notre objectif principal est d'obtenir des résultats sur la représentation et le traitement des variables qui, selon nous, sont fondamentaux. Par ailleurs, aussi bien les observations que nous avons

Educational Studies in Mathematics 16 (1985) 143–161. 0013–1954/85.15.

faites en classe (Rochier, *et al.*, 1984) que les résultats de Soloway et Ehrlich (1982a, 1982b), de Soloway et Bonar (1983) et de Hoc (1978, 1983) montrent que les difficultés majeures des débutants relèvent de la représentation et du traitement des variables. Notre travail s'inscrit dans un cadre théorique qui s'appuie sur l'analyse épistémologique et cognitive des concepts enseignés (Vergnaud, 1983).

2. ELÊMENTS POUR UNE ANALYSE COGNITIVE ET ÉPISTÉMOLOGIQUE DU CONCEPT DE VARIABLE

La notion de variable, au même titre que celle de boucle, intervient comme une notion centrale dans la programmation informatique. La formule "une variable c'est une adresse', ne rend que très partiellement compte du caractère "invariant" que présente, paradoxalement, le concept de variable: en effet, si la valeur d'une variable varie, comme il se doit, non seulement sa désignation mais également sa relation fonctionnelle avec les instructions et les autres variables du programme sont, elles, *invariantes*. Le caractère invariant se manifeste évidemment par la donnée d'un nom à la variable, mais aussi par le contrôle exercé sur l'ensemble des valeurs qu'elle peut prendre et, dans certains types de langages, par la précision d'un type qui détermine les opérations qu'on peut effectuer sur les variables. De ce point de vue, l'étude du concept de variable rejoint des études plus générales de la formation des connaissances et notamment des invariants opératoires.

Il est nécessaire pour les besoins de l'analyse de dissocier les concepts de boucle, de variable et d'affectation, mais ce sont des concepts indissociables du point de vue de processus de conceptualisation par le sujet. Les situations problèmes que recontre le sujet font nécessairement appel à ces concepts dans leur interaction.

Dans la résolution d'un problème informatique, concernant les variables, le sujet est confronté à plusieurs activités distinctes. On peut en distinguer deux types:

- la construction de la signification et des opérations sur les variables: la *déclaration*, l'*affectation* des valeurs, l'*entrée* et la *sortie* des données sur l'écran,
- le contrôle des valeurs particulières qui doivent être prises par les variables lors de l'éxécution du programme: ce contrôle intervient dans la planification des instructions par les structures que sont la *répétition*, le *choix* et la *séquentialité*.

La planification du traitement des variables dans une répétition est de loin l'activité la plus significative dans la conceptualisation des notions de variable et de boucle. C'est surtout dans cette dernière activité que le concept de variable

informatique apparaît différent de celui de variable mathématique (symbole représentant un élément non spécifié ou inconnu d'un ensemble), et que l'opération d'affectation (relation asymétrique) diffère de la relation d'égalité (évidemment symétrique). Dans la structuration d'une boucle les variables interviennent dans trois opérations:

- la mise à jour
- le test d'arrêt
- l'initialisation.

La tâche cognitive du sujet comporte donc trois aspects à la fois:

- mise à jour: identification, élaboration et expression de la règle de définition des variables d'accumulation. Les variables d'accumulation sont celles dont les valeurs sont transformées dans une boucle. C'est le cas par exemple, d'une somme partielle dans une sommation de nombres ou de la variable dite "maximum partie" dans un tri.
- identification du statut fontionnel du test d'arrêt: sur quelle variable porte-t-il? pour quelle valeur? à quel moment intervient-il? en relation avec quel aspect du problème à résoudre?
- initialisation des variables qui sont transformées par l'invariant de boucle.

L'opération de mise à jour s'applique en particulier aux variables d'accumulation qui constituent le type de variable le plus important et le plus nouveau du point de vue conceptuel pour le sujet: elles ne peuvent pas être traitées par un modèle mathématique qui fasse correspondre une et une seule valeur à une variable. Dans l'opération de mise à jour, les aspects qui relèvent de l'algorithmique et de la programmation sont en étroite interaction. Il existe également des relations entre l'initialisation et la mise à jour. Bien qu'il puisse y avoir des exceptions, le plus souvent on initialise une variable par le même moyen que celui qui permet de la mettre à jour: la lecture ou l'affectation.

3. OBJECTIFS

Pour comprendre les difficultés recontrées dans l'enseignement d'une notion, interpéter les procédures mises en oeuvre par les élèves dans la résolution de problème, il faut avoir une image aussi prècise que possible des conceptions des élèves à propos de cette notion. Cela nécessite que soient prélevées, en cours d'apprentissage, des informations relatives aux conceptions des élèves et à leurs évolution.

Quelles conception les élèves de seconde[3] ont-ils de la notion de variable informatique après 15 heures de programmation? Cela conduit notamment à se demander comment ils opèrent sur les différents types de variables qui interviennent effectivement dans la résolution de problème.

Dans les situations-problèmes que rencontre effectivement le sujet, les variables interviennent avec des statuts fonctionnels différents. Dans un premier temps on peut en distinguer deux:

- les variables qui sont les données explicites du problème;
- les variables qui sont rendues nécessaires par la solution informatique. Par exemple, la solution d'un problème d'échange des valeurs de deux variables A et B fait intervenir ces deux types de variables: A et B sont des données du premier type, la variable intermédiaire C qu'il est nécessaire d'utiliser pour mémoriser le contenu d'une des variables au moment de l'échange est du second type.

Par ailleurs, les différentes opérations que le sujet doit faire sur les variables ne sont pas de difficulté cognitive équivalente. La déclaration, l'initialisation, la mise à jour et le test nécessitent non seulement des activités différentes entre elles, mais également en fonction des différentes variables sur lesquelles elles opèrent. Par exemple, le sujet peut repérer correctement l'ensemble des variables simples dans la phase de déclaration sans pour autant se représenter correctement leur fonction. Cette représentation est par contre indispensable pour la réalisation des autres opérations.

Le sujet peut par ailleurs, se représenter la fonction de certaines variables mieux que d'autres, suivant leur signification. Par exemple, les variables qui jouent le rôle de compteur, font appel dans l'opération de mise à jour à des activités cognitives plus faciles que celles qui jouent un rôle de "récapitulation" des résultats:

compteur: = [4] compteur + 1

se réalise par l'ajout d'une constante. La fonction sous-jacente est une fonction de succession et non de sommation comme dans le cas de

some: = somme + nombre

qui indique l'addition à la valeur de la variable d'accumulation de la valeur d'une autre variable.

4. MÉTHODOLOGIE

Les méthodes d'accès aux conceptions des élèves sont très variées. Les conceptions des élèves peuvent être représentées sur deux plans:

- le plan synchronique qui permet de rendre compte de l'état des compétences du sujet à un moment donné de son apprentissage;
- le plan diachronique qui permet de rendre compte de l'évolution des connaissances et des procédures au cours du temps (par exemple au cours d'une ou de plusieurs séances).

Dans ce travail nous nous en tiendrons à une analyse du premier type. Pour le recueil des données correspondant à cette étude nous utilisions une technique qui est fréquemment pratiquée dans ce genre de travail (Bonar *et al.*, 1982): on donne au sujet un texte de programme, on l'informe de la fonction précise du programme et de l'endroit dans le texte où manquent certains éléments. On lui demande alors de compléter les lignes manquantes.

La tâche du sujet n'est pas du même ordre que celle qu'il rencontre lorsqu'il doit lui-même construire un programme: elle contient nécessairement une sous-tâche de lecture et d'analyse d'un texte de programme qu'il n'a pas conçu, tâche sur laquelle on sait actuellement peu de choses. La tâche de construction d'un programme complet reste trop difficile pour des élèves débutants: le nombre élevé de non-réponses ne permet pas de s'appuyer sur des observables suffisamment significatifs.

5. SUJETS

Le questionnaire a été proposé à 26 élèves de la classe de seconde appartenant à des groupes différents: un groupe volontaire ($N = 8$) et deux demi-classes ($N = 10$ et $N = 8$) d'élèves non-volontaires.

Les groupes ont tous suivi un enseignement de la programmation en PASCAL, à raison d'une heure par semaine, au cours duquel ont été introduites les notions suivantes:

- variables: entiers, réels, tableau d'entiers et de réels;
- manipulation des variables: déclaration, expressions arithmétiques, affectation, initialisation etc . . .
- boucle: sous deux formes
  REPEAT ⟨bloc⟩ UNTIL ⟨condition booléenne⟩
  FOR I: = TO N DO ⟨bloc⟩
- instructions de lecture et d'écriture.

Les groupes diffèrent du point de vue de leur niveau de compétence générale en programmation. Nous les noterons respectivement par G1 "bon", G2 "moyen", G3 "faible".

6. ANALYSE DES PROBLÈMES POSÉS ET HYPOTHÈSES

Notre hypothèse générale est que plus les variables à traiter s'éloignent du modèle familier de l'exécution "à la main" qui constitue le traitement du problème par le sujet lui-même, plus leur construction est difficile. C'est le cas notamment pour les variables qui sont constitutives de l'invariant de boucle.

On peut ainsi s'attendre à ce que les variables qui sont des données explicites soient d'une manière générale plus facilement traitées que celles qui sont rendues nécessaires par la solution informatiqu de problème (voir distinction plus haut).

Le questionnaire comporte trois types de tâches (4 questions).

6.1. *Initialisation des variables*

On peut énoncer deux règles à propos de cette opération:

- doit être nécessairement initialisée toute variable dont la valeur est modifiée dans un corps de boucle;
- le mode d'unitialisation est généralement déterminé par le type d'opération utilisé dans la mise à jour.

Toutefois la mise en oeuvre de ces règles ne suffit pas à déterminer la valeur initiale de la variable; celle-ci dépend de l'énoncé de problème et de l'identification des règles de transformation qui opèrent sur la variable.

Le questionnaire comporte trois questions d'initialisation.

(1) La première porte sur trois variables d'un programme, lequel en comporte plusieurs autres (Fig. 1).

Pour répondre à la question, le sujet doit à la fois repérer les variables d'accumulation (COMPTEUR, EXPOX, EXPOY) et identifier leur valeur initiale.

On peut s'attendre à une assez large distribution des réponses ainsi qu'à un traitement particulier de la variable COMPTEUR. Comme nous l'avons souligné plus haut, les élèves ont eu plusieurs occasions de manipuler cette variable. Par ailleurs, l'initialisation du compteur à zéro correspond à l'idée selon laquelle "avant de commencer un calcul relatif à une variable il faut vider la mémoire correspondante"' A ce titre, zéro ne peut pas apparaître comme une valeur particulière du COMPTEUR. L'initialisation à 1 de EXPOX et EXPOY est plus complexe: non seulement on "vide" la mémoire, mais on fixe une valeur particulière de la variable.

(2) La deuxième question (Fig. 2) porte sur l'unique variable d'un programme dans lequel elle intervient avec un double statut: elle est à la fois la donnée explicite et la variable compteur nécessitée par la solution informatique. Mais cette fonction de compteur n'est pas aussi facilement identifiable que pour la variable COMPTEUR de problème précédent.

Pour réussir la tâche il faut coordonner deux informations: celle qui concerne la loi de mise à jour et celle qui concerne la valeur pour laquelle se réalise le test d'arrêt (cela suppose également un accord implicite sur la valeur de la variable qui rend l'expression booléenne VRAIE au moins une fois).

Le programme suivant calcule et affiche $x^n - y^n$ pour x, y, n des entiers positifs.

```
Program expo;
var
   x, y, n, expox, expoy, compteur, resultat: integer;

begin
   read(x);
   read(y);
   read(n);                                  compteur: =0;
#                                              expox: =1;
   repeat                                      expoy: =1;
      begin
         expox: =expox*x;
         expoy: =expoy*y;
         compteur: =compteur + 1;
      end
   until compteur = n;
   resultat: = expox - expoy;
   write ('le resultat est: ', resultat);
end
```

Manque-t-il des instructions là où il y a le, signe #?
oui non
Si oui remettez-les

Note: Les lignes manquantes sont dans la case.

Fig. 1. Problème 1.

La solution canonique du problème est l'initialisation de la variable x par l'affectation d'un nombre impair (cela peut se déduire de la valeur finale de x et de la loi de progression). Cela dit, la réponse qui consiste à initialiser x par la lecture est acceptable à condition que la valeur initiale entrée par l'utilisateur du programme soit toujours un nombre impair.

Selon notre hypothèse générale, on peut s'attendre à une prédominance de l'initialisation par la lecture.

(3) La troisième tâche d'initialisation intervient dans un problème à plusieurs composantes. Nous l'examinerons plus loin dans le cadre du problème 4 (Fig. 4) en relation avec la tâche de mise à jour.

6.2. *Construction du test d'arrêt*

On peut énoncer la règle générale suivante:

Le programme suivant doit afficher une suite de nombres entiers strictement positifs. Une instruction a été effacée (là où il y à le signe #). Remettez-là.

```
program suite entiers;
var

   x:integer;
begin
#                                          x: =1;
   repeat
      begin
         writeln(x);
         x: =x+2;
      end
   until x=32767 (*);
      end
```

(*) 32767 est le nombre entier maximum de la machine

Note: La ligne manquante est dans la case.

Fig. 2. Problème 2.

- une boucle est une opération finie, elle délimite une suite d'actions par un début et une fin.

L'utilisation de cette règle permet seulement de repérer syntaxiquement l'endroit où doit se trouver le test d'arrêt. La tâche de construction du test d'arrêt nécessite en outre l'identification de la variable sur laquelle doit porter le test, et de la valeur particulière que doit prendre cette variable pour que l'expression booléenne puisse prendre au moins une fois la valeur VRAI. Cela suppose la représentation exacte des rôles joués par les différentes variables dans le programme.

Cette tâche intervient dans deux questions:

(1) Dans l'une, la variable sur laquelle il faut opérer a un double statut: elle fonctionne à la fois comme donnée explicite et comme compteur dégressif (Fig. 3).

L'algorithme du calcul de la multiplication avait déjà fait l'objet des premiéres leçons. Dans la version que les élèves avaient recontrés, la variation du compteur se faisait par l'incrémentation avec un test sur l'égalité de la valeur du compteur et de la valeur d'arrêt. Tandis que dans le programme ici, y est une valeur lue, (c'est l'un des termes du produit), et sa veleur est modifiée au cours de l'exécution du calcul.

(2) Dans la seconde question, la variable est un compteur, explicite, mais la construction du test d'arrêt intervient dans une tâche plus complexe (Fig. 4).

Le programme suivant doit calculer et afficher le produit de deux nombres entiers positifs x et y. Une ligne a été effacée (là où il y a le signe #). Remettez-là.

```
program multipl:
var
   x, y, resultat: integer;
begin
   read (x);
   read (y);
   resultat: = 0;
      repeat
         begin
            resultat: = resultat + x;
                  y: = y − 1;
         end
#                                          until y = 0;
            write(resultat);
end
```

Note: Les lignes manquantes sont dans la case

Fig. 3. Problème 3.

On peut s'attendre à ce que la première question soulève plus de difficulté que la seconde.

6.3. *Mise à jour des variables*

Cette tâche intervient dans un problème où il faut construire le corps de boucle d'un programme à partir de la donnée d'un énoncé et d'un programme qui sert de modèle analogique.

Le programme-modèle et le programme à compléter ne diffèrent l'un de l'autre que du point de vue de la gestion de la variable de calcul (RESULTAT). Pour résoudre le problème il faut identifier le rôle des deux variables liées au début du programme, établir la relation qui permet de contrôler le changement de valeur de la variable RESULTAT; il faut initialiser cette variable et déterminer la valeur d'arrêt du compteur. Par rapport au modèle, la seule variation concerne la transformation effectuée sur la variable RESULTAT et l'initialisation de cette variable. En terme de calcul relationnel, les élèves doivent opérer sur quatre relations A1, A2, B1, B2.

A1 "divise un nombre par deux un certain nombre de fois et affiche le résultat final".

```
program partage;
   (*ce programme divise un nombre par deux un certain nombre de fois et
affiche le résultat final; le nombre initial et le nombre de divisions sont choisis
par l'opérateur*)
   var
      nombre, nbrecoup, compteur:integer;
      result:real

   begin
      writeln ('donnez le nombre a diviser');
      read(nombre);
      writeln('donnez le nombre de divisions');
      read(nbrecoup);
      compteur: =0;
      result: =nombre;
         repeat
            begin
               result: =result./.2;
               compteur: =compteur + 1;
            end;
         until compteur=nbrecoup;
      write('le resultat final des divisions est');
      write(result);
   end.
```

En vous aidant de ce programme, complétez le programme suivant (programme triple) qui multiplie par 3 le carré d'un nombre un certain nombre de fois et affiche le résultat final; le nombre initial et le nombre de multiplications sont choisis par l'opérateur.
Rappel: le carré d'un nombre est le produit de ce nombre par lui-même (x * x).

```
program triple;
var
   nombre, nbrecoup, compteur: integer;
   result:real
begin
      read(nombre);
      read(nbrecoup);
      compteur: =0;
      result: =
         repeat
            begin
#
#
            end
until compteur=
      write('le result final est');
      write(result);
end
```

```
nombre*nombre;


result: =result*3;
compteur: =compteur+1;

nbrecoup;
```

Note: Les lignes manquantes sont dans la case

Fig. 4. Problème 4.

TABLEAU I
Réponses au problème 1

– initialisent correctement les trois variables	1
– répondent "non" il ne manque pas d'instruction	4
– répondent "oui" mais n'arrivent pas à identifier les variables	3
– initialisent partiellement	12
– COMPTEUR seulement	7
– EXPOX seulement	1
– EXPOX et COMPTEUR seulement	1
– RESULTAT seulement	3
– initialisation erronnée	4
– read (EXPOX), read(EXPOY)	2
– read (EXPOX)	
– n: =0	
– sans réponse	2

A2: "multiplie par trois le carrée d'une nombre un certain nombre de fois".

B1: "result: = nombre
result: = result/2"

B2: "result: = nombre * nombre
result: = result * 3"

B2 est à B1, ce que A2 est à A1. Il ne s'agit pas d'une simple tâche de reproduction. Ce qu'il s'agit de reproduire, ce n'est pas seulement une forme mais un rapport entre deux énoncés appartenant à des domaines opératoires distincts.

7. ANALYSE DES RÉSULTATS

Nous analysons les résultats en deux parties. Dans la première partie nous examinons les différents types de réponses apportées pour chaque question. Dans la seconde partie nous comparons les résultats en fonction du plan expérimental: types de variables, différentes opérations sur les variables et groupes d'élèves.

7.1. *Analyse par question*

La Tableu I donne la répartition des réponses pour le problème 1 (Fig. 1).

On voit que seulement un élève initialise correctement les trois variables. Les résultats les plus intéressants concernent l'initialisation partielle. Conformément à l'analyse faite plus haut, les variables COMPTEUR et RESULTAT

jouent un rôle privilégié. La variable COMPTEUR est initialisée par 3 élèves alors que ce n'est pas nécessaire. Ce sont bien les variables d'accumulation qui posent le plus de problèmes aux élèves. Cette difficulté concerne autant le rôle de l'initialisation dans la définition d'une variable que la représentation de la fonction de cette variable dans le calcul. L'initialisation faite à l'aide d'instructions de lecture (3 élèves) témoigne d'une opinion exprimée par les élèves selon laquelle: "*on doit dire à la machine les valeurs des variables sinon elle ne comprendra pas ce que sont les noms*". D'ailleures, ces élèves n'initialisent pas le COMPTEUR.

Le Tableau II donne la répartition des différentes réponses pour le deuxième problème (Fig. 2).

TABLEAU II

Réponses au problème 2

initialisation	
- par lecture "read(x)"	16
- par affectation	6
-x: =0	5
-x: =1	1
- par lecture et affectation	2
- sans réponse	2

Conformément aux hypothèses que nous avions formulées plus haut, les résultats indiquent une nette prédominance de l'initialisation par lecture sur l'initialisation par affectation. Dans ce dernier cas, l'initialisation à zéro est la plus fréquente, ce qui confirme l'analyse "a priori" que nous avions faite. On peut signaler également le cas des deux élèves qui initialisent à la fois par une instruction de lecture et par une affectation: comme si la lecture ne suffisait pas à donner une valeur à une variable. Cette conduite pourraît être vue comme la séquelle d'une conception initiale selon laquelle, la lecture

Read (x)

ne fait que *prendre* de l'écran la valeur de la variable x, mais ne permet pas d'affecter cette valeur à l'adresse indiquée par x: le sujet complète alors lui-même cette seconde fonction, en utilisant l'affectation en plus de la lecture.

Les réponses données au troisième problème (Fig. 3) figurent dans le Tableau III.

Les résultats permettent d'établir quatre types de réponses hiérarchisés. Le premier correspond à la réponse exacte: le test d'arrêt porte sur la variable y. Dans un cas sur 11 réponses de ce type, l'erreur porte sur la valeur de la variable y: elle est égale à 1.

Le deuxiéme type de réponses correspond au modéle selon lequel "la

TABLEAU III
Réponses au problème 3

– le test d'arrêt porte sur une valeur particulière de la variable y	11
until y = 0	10
until y = 1	1
– répéter jusqu' à ce que le résultat soit le produit	3
until y * x resultat = y * x resultat = resultat + x * y − 1	
– répéter jusqu' à ce que le résultat soit obtenu	3
until y = résultat resultat resultat différent de 0	
– fin d'instruction "until" sans test	8
– sans réponse	1

répétition s'arrête lorsque le produit $x * y$ est obtenu". Il y a trois réponses de ce type 1.

Le troisième type de réponses correspond à un modèle plus primitif que les deux précédents: "la répétition s'arrête lorsque le résultat est obtenu", sans que soit précisée la nature exacte du résultat.

Enfin le quatrième type de réponses et caractérisé par l'utilisation seule des indices syntaxiques: "à chaque REPEAT (début d'instruction) on associe UNTIL (fin d'instruction)". D'ailleurs, cette association est bien établie par la majorité des élèves (22 sur 26).

Enfin le Tableau IV donne les réponses obtenues au problème 4 (Fig. 4).

On peut constater que 11 sujets sur 26 complétent correctement le corps de boucle, c'est-à-dire réussissent la tâche de mise à jour. Il y a deux cas d'initialisation erronée de la variable RESULT:

- pour l'un des sujets "nombre^2" correspond à une simple erreur de syntaxe, le langage PASCAL ne permettant pas de notation exponentielle;
- pour l'autre sujet, outre l'erreur syntaxique, le changement de dénomination de la variable correspond à une difficulté à faire intervenir la variable NOMBRE dans le calcul.

En général, aussi bien au niveau des procédures réussies qu'au niveau des éches, il y a une dissociation nette des deux variables RESULT et COMPTEUR. Les difficultés portent essentiellement sur la variable d'accumulation elle-même.

Il ya a deux types de réponses qui représentent à eux seuls plus de la moitié des cas. D'une part la réponse juste, d'autre part une réponse dans laquelle les sujets reproduisent simplement l'instruction d'initialisation du programme modéle et proposent une instruction de mise à jour de la varibale RESULT qui

TABLEAU IV

Réponses au problème 4

- procédure et expression correctes 9
- procédure correcte avec erreur d'expression en PASCAL 2
  result: = x^2
  result: = nombre^2
- erreur d'initialisation et de traitement de l'affectation comme une égalité 5
  result: = nombre
  repeat
  begin
  result: = result * result * 3
  compteur: = compteur + 1
  end
  until compteur = nbrecoup
- calquage du modèle sans modification de procédure 3
  - result: = result/3 2
  - result: = nombre
  result: = result*3
- "solution algébrique" traitement de boucle comme une série d'operations algébriques 3
  - result = x^2*3
  - result: = (3*(nombre*nombre))*nbrecoup
  - repeat
  result: = nombre*nombre
  result: = result*3
- autres 3
  - repeat x*x
  - result: = nombre
  repeat
  result: = nombre
  - result: = nombre*nombre
  repeat
  result: = nombre*3
- sans réponse 1

tient compte de cette initialisation. Une des interprétations possibles est que les sujets traitent l'affectation comme une relation d'égalité et que le programme représente pour eux une suite d'expressions algébriques. Le programme ainsi complété est juste *pour la valeur 1 de la variable NBRECOUP.*

Il y a trois procédures qui consistent à reprendre avec un minimum de modification les éléments correspondants du programme modèle. Il est vrai que nous ne nous étions pas donné les moyens, dans ce questionnaire, de voir dans quelle mesure l'écriture des relations algébriques correspondant aux énoncés pouvait intervenir dans la détermination des instructions de changement de valeur des variables d'accumulation. Lors de la séance de correction du questionnaire, l'énoncé "multiplier par 3 le carré d'un nombre, un certain nombre de fois" a donné lieu à plusieurs formulations et écritures:

- $3x^2$ par le nombre de coup
- $3x^2y$ (pour $(3x^2)\,y$ fois)
- $(x^2y)\,3y$
- $x^2 \underbrace{\times 3 \times 3 \times 3 \times \ldots \times 3}_{y \text{ fois}}$ (traduit par x^2y^3)

avant de parvenir à la formule $x^2 3^y$.

Les difficultés à formuler dans un sytème d'écriture de type algeébrique la fonction décrite par l'énoncé sont bien connues (Ehrlich *et al.*, 1982). On comprend de ce fait le nombre relativement élevé de procédures erronées. Sans entrer dans le détail, on peut estimer que l'instruction de changement de valeur de la variable d'accumulation demande une représentation fonctionelle particulière de la relation décrite par l'énoncé. Il ne faut pas ici considérer l'aspect purement algébrique, mais plutôt concevoir la formule comme une image de la succession des calculs nécessaires:

$$\text{de } x^2 3^p \quad \text{à} \quad (\ldots(((x^2 \times 3) \times 3) \times 3) \ldots \times 3).$$

Il est donc indispensable de parvenir à une lecture algorithmique des formules algébriques. Dans le probléme que nous venons d'etudier, se dessine assez nettement l'idée qu'on peut se faire des difficultés que recontrent les élèves. C'est un point qui mérite des études complémentaires.

7.2. *Analyse transversale: conceptions à propos de la notion de variable*

Du point de vue opératoire, c'est-à-dire du point de vue des actions à entreprendre et des décisions à prendre par le sujet lors de la construction d'un programme correspondant à un problème donné, nous avons distingué au début de cet article deux types de variables:

- les variables qui correspondent à des données explicites du problème;
- les variables qui interviennent dans la solution informatique.

Dans notre questionnaire les décisions à prendre à propos de ces variables, concernent les opérations suivantes:

- initialisation;
- mise à jour;
- test d'arrêt.

On peut alors se demander, d'une part si ces deux types de variables sont traitées de la même manière par le sujet, d'autre part quelles sont les opérations les plus facilement mobilisables dans la construction d'un programme.

Le Tableau V donne la répartition des réussites en fonctions des caractéristiques fonctionnelles des variables, des types d'opérations demandées, et des différents groupes examinés.

TABLEAU V

Répartition des réussites en fonction des caractéristiques fonctionnelles de variables

Operations	Types de variables		Groupes G1 $N=8$	G2 $N=10$	G3 $N=8$	Total $N=26$
	Variable d'accumulation non donnée du problème		1	-	-	1
Initialisation	Variable compteur		3	3	3	9
	Variable donnée unique au problème	par lecture	4	7	5	16
		par affectation	4	1	1	6
Mise à jour	Variable compteur		8	8	6	22
	Variable d'accumulation		5	2	5	12
Test d'arrêt	Variable compteur explicite		8	8	4	20
	Variable compteur		7	3	-	10

Note: G1, G2, G3 dénotent respectivement les groupes "bon", "moyen" et "faible" du point de vue de leur niveau général.

On note que la tâche d'initialisation est globalement celle qui pose le plus de problèmes aux élèves. On peut estimer d'ailleurs que les élèves recontrent ici une classe de problèmes qui soulève des difficultés d'orde général dans de nombreux autres domains (Vergnaud, 1982): *faire une hypothèse sur l'état initial d'une variable, connaissant la transformation et l'état final.*

Par ailleurs, on remarque que, des deux modalités d'initialisation de certaines variables, affectation et lecture, c'est la lecture qui est privilégiée. Cela peut s'expliquer par deux arguments:

- la relation entre l'instruction et son effet peut être plus facilement établie dans le cas de la lecture puisque le sujet est impliqué lui-même au cours de l'exécution du programme: "il entre des données au clavier";
- le sujet fait fonctionner probablement une règle implicite selon laquelle "chaque programme doit comporter nécessairement une instruction de lecture pour pouvoir opérer sur des valeurs particulières".

Les opérations de mise à jour et de test d'arrêt semblent être de difficulté équivalente.

En ce qui concerne le traitment des différents types de variables, on note que la variable COMPTEUR est toujours mieux traitée que les autres variables lorsqu'elle obéit au modèle canonique: "elle s'initialise à zéro, elle s'incrémente de 1". Lorsque la variable qui joue le rôle de compteur ne correspond pas explicitement à ce modèle, les réussites chutent de moitié comme par exemple dans la tâche de construction du test d'arrêt. Les variables d'accumulation de résultats intermédiaires sont dans tous les cas, plus difficiles à traiter.

Quant aux différents groupes examinés, on remarque que les tâches les plus

difficiles sont également les moins bien réussies par le groupe "failble". Cela s'observe en particulier dans les tâches d'initialisation et de construction du test d'arrêt. La différence entre les groupes d'élèves se manifeste en particulier dans le traitement des différents types de variables. Le groupe "faible" a plus de difficulté que les autres groupes à se représenter le rôle fonctionnel des variables d'accumulation et les traitements qui leur sont associés.

8. CONCLUSIONS ET DISCUSSION

Le travail que nous avons présenté a pour but l'etude du fonctionnement de la notion de variable informatique chez des élèves débutants qui ont suivi une séquence didactique de 15 heures.

Nous avons proposé aux élèves un questionnaire qui a été conçu à partir des hypothèses suivantes sur le fonctionnement, dans la résolution de problèmes, des différents types de variables et d'opérations sur ces variables:

- les données explicites sont des informations directement repérables, elles seront mieux traitées que les autres variables;
- les variables institutionnalisées[5] par l'enseignement de la structure de boucle seront également mieux traitées.
- l'initialisation sera une tâche difficile pour l'ensemble des élèves.

Les résultats essentiels peuvent être résumés en trois points:

- les élèves font une meilleur utilisation et un meilleur traitement des variables instituionnalisées dans l'enseignement (par exemple le compteur);
- l'initialisation des variables est une opération cognitive difficile: des deux modalités d'initialisation c'est la lecture qui joue un rôle privilégié;
- les difficultés présentées par la construction d'une expression booléene qui intervient dans un test d'arrêt et par la construction d'un corps de boucle sont du même ordre.

Il est clair que nous ne saurions réduire les problèmes de conceptualisation de la notion de variable à la seule question des modalités sous lesquelles elle intervient dans un programme. Néanmoins il s'agit là d'une question importante que nous avons examinée à travers ce questionnaire. Il s'agit d'une première approche des problèmes liés aux manipulations de variables et nous pensons que ces résultats sont significatifs, au moins pour la période d'"alphabétisation".

Concernant les aspects que nous avons étudiés, *certaines des conceptions des élèves peuvent se résumer de la manière suivante:*

(1) "un programme opére sur des valeurs particulières. Ces valeurs ne peuvent être communiquées à l'ordinateur que par une instruction de lecture".

Cela explique d'une part le privilège accordé à la lecture comme modalité d'initialisation et d'autre part l'utilisation simultanée des instructions de lecture et d'affectation pour une même variable. On peut alors penser que le sujet comprend d'abord les instructions interactives d'entrée-sortie.

(2) "au début de chaque programme il faut vider les cases mémoires associées aux variables".

Cette conception explique l'initialisation à zéro de l'ensemble des variables indépendamment de leur mode d'évolution ultérieur:

(3) "le compteur s'initialise à zéro, s'incrémente de 1".

Cette conception permet de traiter correctement les variables qui correspondent au modèle canonique.

En ce qui concerne le test d'arrêt, les résultats permettent de distinguer plusieurs conceptions que nous avons décrites plus haut (Tableau III). Lorsque le sujet n'arrive pas à construire le test d'arrêt en coordonnant les éléments pertinents du problème, il met en oeuvre les conceptions primitives suivantes:

- "l'instruction de répétiition est construite par deux termes REPEAT et UNTIL";
- "la répétition s'arrête lorsque le résultat est obtenu".

Nous avons soulevé ainsi plusieurs sortes de problèmes à propos des conceptions primitives des élèves concernant les notions fondamentales de variable, de boucle, d'entrée et de sortie.

Il nous paraît indispensable que des études plus nombreuses et plus complètes soient poursuivies à la fois pour comprendre les mécanismes d'acquisition des notions ci-dessus et pour mettre en place des moyens didactiques de surmonter les difficultés recontrées par les élèves.

NOTES

[1] Cette recherche a été réalisée en collaboration avec A. Rouchier, J. Rogalski, J. C. Despland, J. M. Laubin, Y. Ferrand, C. Landré, J. F. Pigeonnat, G. Sarfati, et G. Vergnaud.

[2] L'alphabétisation: La toute première péroide de l'initialisation à l'informatique.

[3] Cinquième année de l'enseignement secondaire français.

[4] Le signe: = dénote l'affectation en PASCAL.

[5] L'institutionnalisation caractérise tout procédé qui vise à donner à une connaissance (concept ou procédure) le statut d'un savoir à retenir.

REFERENCES

Anderson, J. R., Farell, R., and Sauers, R.: 1984, 'Programming in LISP', *Cognitive Science* 8, 87–129.

Bonar, J. P., Ehrlich, K., Soloway, E., and Rubin, E.: 1982, 'Collecting and analyzing on-line protocols from novice programmers', *Behaviour Research Methods and Instrumentation* **14**, 203–209.

Ehrlich, K., Soloway, E., and Abbot, V.: 1982a, *Transfer Effects from Programming to Algebra Word Problems: a Preliminary Study*, Research Report, Dept. of Computer Science, Yale University.

Ehrlich, K. and Soloway, E.: 1982b, *An Empirical Investigation of the Tacit Plan Knowledge in Programming*, Technical Report, Dept. of Computer Science, Yale University.

Hoc, J. M.: 1978, *La programmation informatique comme situation de résolution de problème*, Thèse de 3ème cycle, E.P.H.E., Paris.

Hoc, J. M.: 1982, 'Analysis of beginners' problem-solving strategies in programming', in T.R.G. Green and G. Van der Veer (eds.), *The Psychology of Computer Use*, Academic Press, London.

Jeffries, R.: 1982, 'A comparison of the debugging behaviour of expert and novice programmers', in *The Proceeding of AERA Annual Meeting.*

Rouchier A., Samurçay, R., Rogalski, J., Despland, J. C., Laubin, J. M., Landré, C., Pigeonnat J. F., Sarfati, G., Ferrand, Y., and Vergnaud, G. 1984, *Concepts informatiques et Programmation. Une première Analyse en classe de seconde des lycées*, Rapport de Recherche, CNRS-Université d'Orléans.

Shneiderman, B.: 1980, *Software Psychology, Human Factors in Computer and Information Systems*, Winthrop Publishers, Inc', Cambridge.

Soloway, E., Ehrlich, K., Bonar, J., and Greenspan, J.: 1982a, 'What do novices know about programming?', in B. Shneiderman et A. Badre, (eds), *Directions in Human–Computer Interactions*, Ablex Publishing Co.

Soloway, E., Bonar, J., and Ehrlich, K.: 1982b, 'Cognitive strategies and looping constructs: an empirical study', in *The Communications of ACM.*

Vergnaud, G.: 1982, 'A classification of cognitive tasks and operations of thought involved in addition and subtraction problems', in T. P. Carpenter, J. M. Moser, and T. A. Romberg (eds.), *Addition and Subtraction: A Cognitive Perspective*, Hillsdale, Lawrence Erlbaum.

Vergnaud, G.: 1982, 'Why is an epistemological perspective a necessity for research in mathematics eductation?', in *proceedings of the Fifth Annual Meeting PME-NA*, 2–20.

Centre d'Etude des Processus,
Cognitifs et du Langage,
CNRS-EHESS, 54, Bvd Raspail,
75270 Paris Cedex 06, France

English title:

THE CONCEPT OF VARIABLE IN PROGRAMMING –
ITS MEANING AND USE IN PROBLEM-SOLVING