Chapitre 3 Cadre général de recherche

à quel public s'intéresser?
le sujet confronté à l'activité de programmation
le problème de programmation

Ce chapitre présente le cadre général d'hypothèses dans lequel cette recherche s'insère. Ce cadre prend en compte l'analyse des contenus d'enseignement du chapitre 1, et l'analyse de travaux de psychologie de la programmation et de didactique de l'informatique du chapitre 2. Il tente de répondre aux premières questions assez générales, qu'on se pose en début de recherche: une investigation au niveau des débutants étant imposée par l'état actuel de la recherche, il nous faut faire des hypothèses sur le fonctionnement cognitif de ce public face à l'activité de programmation, discuter ce qu'est, à ce niveau, un problème de programmation faisant intervenir les concepts de base de l'informatique et compatible avec les langages accessibles aux débutants.

1. Dans l'état actuel de la didactique de l'informatique, la recherche doit s'intéresser prioritairement à un public hétérogène sans finalité professionnelle

Le choix d'un niveau d'enseignement de l'informatique auquel s'intéresser est la première question que doit se poser un chercheur en didactique tentant de mieux comprendre le fonctionnement cognitif de sujets confrontés aux apprentissages en informatique. Dans l'état actuel de l'enseignement de cette discipline, de nombreux cursus s'adressent à des publics assez spécifiques: étudiants persévérant en informatique dans le cadre d'un enseignement à finalité professionnelle (études professionnelles d'informatique, écoles spécialisées) ou étudiants abordant l'informatique comme complément à une solide formation de base dans le domaine scientifique (écoles d'ingénieur, premiers cycles scientifiques des universités, seconds cycles universitaires d'informatique). La didactique de l'informatique a fait l'objet, jusqu'ici, de peu de recherches et donc, dans une observation portant sur ces publics si particuliers, la part des spécificités des étudiants et de l'enseignement risque d'être difficile à faire. Pour commencer à préciser un cadre d'analyse général, il vaut mieux partir de l'observation d'un public d'élèves n'ayant pas ces spécificités: il sera plus facile de situer ensuite, dans ce cadre général, les particularités de publics spécifiques[53] . C'est pourquoi nous avons fait le choix d'observer des élèves non sélectionnés pour leurs acquisitions antérieures, dans le cadre d'un enseignement sans vocation professionnelle. L'option informatique des lycées que nous présenterons au chapitre 4, nous a paru un terrain d'observation adapté, surtout en première année (classe de Seconde), où le public est le plus hétérogène. Ce travail se situe donc dans une perspective proche de travaux tels que [SAMURCAY 85] et [COHORS-FRESENBORG 87], plutôt que de travaux de psychologie de la programmation tels que [GREEN 77] et [HOC 77]. Néanmoins, l'ensemble de ces travaux, présenté au chapitre **2** va nous être

[53] Au niveau de publics spécifiques, une question souvent posée est celle de l'acquisition de capacités à structurer un programme par une analyse préalable. Il nous semble difficile, dans l'état actuel des recherches en didactique de l'informatique, de savoir si les réticences ou difficultés des étudiants à mener une telle structuration sont liées à des conceptions propres à cette structuration, ou se rapportent à des conceptions erronées plus profondes des objets de base de l'informatique. Il nous semble que de nombreuses études proposant ou discutant des stratégies pédagogiques dans ce domaine manquent d'une étude préalable situant les conceptions et les connaissances des étudiants concernés. Voir par exemple «Une initiation à l'analyse descendante» C. GEOFFROY Actes du premier colloque de didactique de l'Informatique éditions EPI [GEOFFROY 88].

utile pour préciser un cadre général d'analyse des conceptions du sujet face à une tâche de programmation.

2. Le sujet confronté à l'activité de programmation: représentations du dispositif et relation aux concepts

2.1 Les dispositifs informatiques; représentations matérielles, traitements et données

- De [HOC 77], nous avons retenu que les **dispositifs**, qu'ils existent réellement ou qu'ils soient virtuels, sont représentatifs d'unités matérielles agissant sur l'environnement selon des régles entièrement énonçables, et sont organisés selon des hiérarchies d'abstraction: par exemple, un langage de programmation "*évolué*" comme le sous-ensemble de PASCAL utilisé par les élèves observés dans [SAMURCAY 85] diffère de la "*machine à registre*" présentée dans [COHORS-FRESENBORG 87] en ce sens que des éléments du langage (affectation, opérations arithmétiques...) sont primitifs dans le sous-ensemble de PASCAL alors qu'ils doivent être *construits* dans le langage de la "machine à registre". De même le sous-ensemble de PASCAL contient des "avertisseurs d'itération" permettant de composer un calcul itératif à partir de sa définition, sans se préoccuper de la façon dont le dispositif gère les ruptures de séquentialité[54], alors que dans le langage de la "machine à registre", le programmeur dispose seulement d'une forme élémentaire de conditionnelle de sortie d'itération et doit donc chercher une adaptation de la définition du calcul itératif compatible avec les capacités d'expression du langage de la "machine à registre".
- Un dispositif est susceptible de "**représentations matérielles**" diverses: la relation entre le dispositif et sa "représentation" matérielle peut être entièrement explicite dans le cas de la "machine à registre" (représentation à l'aide de boîtes d'allumettes, de circuits de chemin de fer...); elle l'est beaucoup moins dans le cas du sous-ensemble de PASCAL, celui ci étant relié à une unité matérielle par un ensemble hiérarchisé de couches logicielles (langage machine, système d'exploitation, micro-code du processeur..) qui ne se laisse pas décrire simplement.
- Dans le dispositif informatique, nous distinguerons d'une part la **structuration des données**, et d'autre part les **traitements** que le dispositif réalise sur ces données. Nous avons vu au chapitre 1 que les notions liées à la structuration des données se sont dégagées progressivement au cours de la jeune histoire de l'informatique, alors que les notions liés au traitements (théorie des algorithmes...) sont présentes dès le départ de l'informatique. Une structuration des données se caractérise comme un "**type abstrait**", c'est-à-dire comme un système formel (une construction axiomatique) faisant intervenir des fonctions sur les objets considérés et leurs propriétés. La "**représentation**" d'une structure dans une autre est un lien abstrait entre les types, analogue à la "*construction* " d'une structure axiomatique dans une autre en mathématiques.
- les structures de données seraient seulement des structures axiomatiques si elles n'étaient pas associées à des traitements dans la **résolution de problèmes de programmation**: la résolution des ces problèmes fait progresser de pair la construction d'algorithmes et la structuration des données, celles ci étant considérées alternativement comme types abstraits et à l'aide de leurs représentations dans d'autres structures.

2.2 Les S.R.T. comme structures mentales locales: éléments relatifs aux données et éléments relatifs aux traitements

Nous avons rencontré dans [HOC 77] les **Systèmes de Représentation et de Traitement (S.R.T.)**: un **S.R.T.** est le produit, chez le sujet, de l'intériorisation d'un domaine de tâches: il permet à un sujet de se représenter un tâche donnée

[54] En fait, ceci n'est possible, dans ce langage que pour une certaine classe d'itérations. Nous approfondirons cette question au chapitre 5.

effectuable sur un dispositif donné, et de faire fonctionner mentalement ce dispositif pour cette tâche. Les **S.R.T.** d'un sujet sont organisés hiérarchiquement selon une relation d'abstraction, et la résolution d'un problème de programmation consiste pour le sujet à élaborer des codes lui permettant de passer d'un **S.R.T.** où il peut se représenter une solution, à un **S.R.T.** correspondant au dispositif; il y a donc, au cours de l'activité de résolution, création par l'élève de codes mentaux reliant différents **S.R.T.**

Un **S.R.T.** donné chez un sujet donné est donc pour nous la structure mentale qui lui permet à un moment précis d'une tâche précise de construire une solution ou de vérifier sa validité. Les conduites que l'on peut observer chez le sujet découlent du **S.R.T.** qu'il fait fonctionner, mais, plus qu'un ensemble de régles de conduite, le **S.R.T.** est une véritable "machine mentale", que le sujet peut faire fonctionner sans que, bien sûr, les régles de fonctionnement soient explicitées ni entièrement explicitables. Le(s) **S.R.T.** que le sujet met à contribution pour la résolution de problèmes de programmation présentent une organisation analogue à celle dispositif pour lequel il programme. C'est pourquoi un **S.R.T.** correspondant à un dispositif informatique contiendra de façon organisée, d'une part certains éléments propres aux **objets** eux-mêmes, et d'autre part des éléments relatifs aux **traitements**.

- Dans le domaine des **objets**, le **S.R.T.** utilisé par le sujet lui permet, par exemple, de distinguer ou non des objets d'un type particulier, d'intégrer ou non la variabilité des objets, la possibilité de les nommer, de faire changer leur valeur; de disposer ou non d'un langage sur ces objets permettant d'exprimer et de combiner les fonctions propres à ces objets. Nous dirons que ces éléments s'organisent, au sein du **S.R.T.**, **en plan(s) relatif(s) aux objets**[55]. Quand, par exemple, un sujet confronté à la complétion d'un dessin en "géométrie-tortue", doit calculer l'angle de rotation nécessaire pour "positionner" la tortue dans la direction qui lui permettra de terminer le dessin, il prend en compte la position atteinte, l'orientation de la tortue, les propriétés géométriques des dessins élémentaires... Il met donc en oeuvre des éléments de représentation mentale de la tortue et du dessin élémentaire à réaliser et des procédures mentales, qui constituent pour nous le plan relatif à l'objet "tortue" du **S.R.T.** que le sujet utilise.
- Le **S.R.T.** permet d'autre part au sujet de se représenter, de planifier, de faire fonctionner mentalement des **traitements complexes**, et donc d'articuler les actions élémentaires sur les objets. Les *plan(s)* relatif(s) aux traitements dans un **S.R.T.** comprennent des représentations des capacités de traitement (réelles ou supposées) du dispositif, des significations que le sujet attribue aux divers éléments syntaxiques marquant le traitement, des procédures d'exécution mentale des traitements complexes... En reprenant l'exemple ci-dessus, le sujet confronté à la complétion d'un dessin en "géométrie-tortue" ne doit pas seulement imaginer comme nous l'avons vu, des actions élémentaires comme les rotations de la tortue, et ses déplacements. En s'appuyant sur sa conceptions de l'objet "tortue", il doit également trouver dans son **S.R.T.** les éléments pour imaginer et coordonner les déplacements de la tortue: selon ses capacités, il trouvera seulement des représentations et des procédures mentales directement déduites de l'expérience sensible (par exemple, il pourra se représenter le tracé d'un carré comme une suite d'actions simples non organisées en répétition), ou il pourra intégrer des représentations plus ou moins adéquates de processus répétitifs avec contrôle, il pourra penser ces processus en termes d'exécution (suite ou répétition d'action) ou en termes de transformation de situations (situation initiale, situations intermédiaires, situation finale..).

Les dispositifs les plus abstraits (par exemple celui que constitue un langage de programmation "évolué"), permettent en premier lieu de considérer toute manipulation sur des objets comme un *calcul* (c'est à dire un processus `données→ résultats`, entièrement explicitable à l'aide de règles formelles) et nous avons vu au chapitre 1 que ceci conduit à considérer d'une part une définition abstraite des objets, et d'autre

[55] En fait, ces conceptions s'organisent en sous-systèmes du **S.R.T.** relatifs chacun à un type de données.

part des représentations qui relient entre elles les structures abstraites. Ces dispositifs comprennent également des traitements généraux indépendants des objets sur lesquels ils opèrent: c'est ainsi que par exemple, la concaténation d'un caractère à une chaîne est un calcul de même nature que l'addition de deux nombres, et le "retournement d'une chaîne de caractères" est une itération isomorphe à l'exponentiation obtenue comme multiplication itérée. Au contraire, dans le **S.R.T.** d'un sujet confronté à une tâche nouvelle de programmation, les éléments relatifs aux objets peuvent rester marqués par une conception *pré-calculatoire* (par exemple, la concaténation peut être intégrée comme un déplacement effectif de caractères), ou les représentations peuvent ne pas être distinguées des définitions abstraites (par exemple, un nombre pourra ne pas être distingué de la chaîne de ses chiffres en base 10) et les éléments relatifs aux traitements peuvent être conçus de façon très dépendante des objets sur lesquels ils portent, ce qui se traduit, par exemple, par le fait que le sujet peut ne pas transférer d'un domaine d'objets à l'autre ses capacités algorithmiques. Nous verrons au §3.1 par quels mécanismes se constitue un **S.R.T.** de ce type.

2.3 Le niveau des concepts informatiques; conséquences pour la stratégie d'apprentissage

Les **S.R.T.** ont été introduits par J.M. HOC pour rendre compte des conduites du sujet face à un ensemble varié de tâches, allant de la conduite de dispositifs matériels à l'utilisation de langages de programmation évolués. L'enseignement de l'informatique, ne vise pas la simple maîtrise des tâches, mais, à travers cette maîtrise, la possibilité de raisonner au niveau des concepts informatiques, et plus seulement au niveau des domaines de tâches. Nous considérerons que la disponibilité pour un sujet d'un concept informatique, se traduit en premier lieu par des capacités de transfert, de différenciation, de coordination d'un **S.R.T.** à l'autre; ainsi, la disponibilité du concept de variable permettra à un sujet de faire le lien entre l'affectation dans un langage impératif comme PASCAL, et l'inscription d'une donnée ou d'une formule dans une cellule d'un tableur comme **Multiplan**, de distinguer une affectation initiale, faite une seule fois dans un programme, d'une affectation se répétant dans le corps d'une itération. En tant qu'exemple de raisonnement au niveau des concepts, l'étude de la complexité d'un algorithme de tri suppose de s'abstraire des objets sur lequel opère l'algorithme, et de la règle de comparaison qui fonde le tri: la complexité prend en compte le nombre d'affectations et/ou de comparaisons effectuées, indépendamment des objets sur lesquels porte l'affectation et indépendamment du type de comparaison. Le niveau des concepts permet donc aussi au sujet de raisonner sur un algorithme ou une structure de données indépendamment du contexte dans lequel ils ont été construit. En particulier, et par différence avec le niveau d'un **S.R.T.** élaboré pour une tâche de programmation particulière, le niveau des concepts est celui où le sujet peut raisonner sur la structuration des données indépendamment d'une structure algorithmique, et sur la structure algorithmique indépendamment de la structuration des données.

Comment un sujet peut-il accéder au niveau des concepts informatiques de base? Mener une étude didactique auprès de débutants, suppose que le chercheur annonce ses hypothèses en matière de stratégie pédagogique. Nous considérons d'abord que le champ des concepts de base en informatique est constitué de notions liées fortement (notion de variable, de traitement conditionnel, d'itération, de structuration des données...), au point qu'il n'est guère possible de définir l'une sans les autres et que par conséquent, des stratégies pédagogiques qui viseraient à présenter ces notions a priori et de façon séparée videraient ces concepts de leur sens, réduisant l'enseignement à l'apprentissage d'un langage. De même, une présentation des algorithmes indépendamment des données ou l'inverse, empêcherait, que ces notions prennent leur sens. Il est donc nécessaire, suivant l'analyse de [BROUSSEAU 86], déjà citée au chapitre 1, que l'enseignant se livre à un travail de *recontextualisation* , «*inverse* » de celui du savant : «*il doit produire une* ***recontextualisation*** *et une* ***repersonnalisation*** *des connaissances*», de façon à ce que les connaissances deviennent «*la connaissance de l'élève, c'est-à-dire une réponse assez naturelle à des conditions (...)indispensables pour qu'elles aient un sens pour lui*». Le travail des

élèves est de «***redécontextualiser*** *et* ***redépersonnaliser*** *leur savoir, (...) de façon (à l') identifier avec le savoir qui a cours dans la communauté scientifique et culturelle de leur époque*». Comme nous l'avons vu au chapitre 1, la (courte) tradition de l'enseignement de l'informatique opère la recontextualisation des concepts par le choix de "**problèmes de programmation**". Un "problème de programmation" se définit comme l'énoncé d'une tâche qu'il s'agit de faire réaliser à un dispositif informatique; de façon générale, le travail de l'élève consiste donc, à partir d'une machine universelle, à réaliser une machine adaptée à une tâche donnée. Mais bien sûr, comme dans un problème de mathématiques, la façon d'obtenir la solution, la discussion des solutions possibles est plus importante que la solution elle-même; nous avons vu au chapitre 1, avec [ARSAC 83] et [PAIR MOHR SCHOTT 88], des exemples d'une telle recontextualisation.. Nous en déduisons que, au niveau des premiers apprentissage, une stratégie pédagogique doit viser en premier lieu la constitution de **S.R.T.** locaux, construits en réponse à des problèmes de programmation où les concepts fondamentaux interagissent, permettre ensuite la mise en relation et la structuration de ces **S.R.T.**, et proposer des situations conduisant les étudiants à considérer ces concepts en dehors des contextes.

Notre recherche portant sur les premiers apprentissages, son sujet principal est par conséquent l'étude de la constitution de ces **S.R.T.**, et il en résulte que le choix des types de problèmes proposés aux étudiants, en particulier la façon dont les concepts sont mis en jeu dans ces problèmes, est fondamental. Nous discutons ce choix au paragraphe suivant.

3. A partir de quels problèmes de programmation observer et faire évoluer les représentations mentales des élèves ?

En fonction des hypothèses énoncées ci-dessus concernant le fonctionnement mental du sujet confronté aux apprentissages en informatique, nous présentons d'abord un schéma de l'activité de programmation que nous pensons adapté au débutant, puis nous discutons plusieurs conceptions des problèmes de programmation, à partir d'un choix d'auteurs, et, enfin, nous exposons les principes qui nous guident dans la conception des problèmes dont nous étudierons la résolution aux chapitres suivants.

3.1 Les problèmes de programmation dans les premiers apprentissages et la constitution des S.R.T.

La figure ci-dessous décrit l'activité de programmation telle que nous la concevons dans le cadre de l'alphabétisation informatique.

<table>
<tr><th colspan="2">Activité de Programmation</th><th>Sujet débutant en Programmation</th></tr>
<tr><td>Connaissances données</td><td>≈≈≈≈≈≈≈≈ Connaissances résultats</td><td>Situation connue (familière)</td></tr>
<tr><td>↓ codification ↓</td><td>↑interprétation↑</td><td></td></tr>
<tr><td>Informations données</td><td>⟶ Informations résultats
traitement algorithmique</td><td>Spécificités, contraintes, rencontrées et intégrées progressivement.</td></tr>
</table>

En partie gauche de la figure, un schéma de l'activité de programmation tiré de [ARSAC

86][56]. Ce schéma nous paraît bien décrire la résolution de problèmes simples, et le fonctionnement de débutants qui sont très marqués par une conception intuitive (naïve) des objets et des traitements. [ARSAC 86] introduit ce schéma pour présenter les implications culturelles qui pourraient résulter selon lui d'une alphabétisation massive en informatique: la résolution d'un problème de programmation suppose que le sujet se représente des "*connaissances données*" et des "*connaissances résultats*"; le plus souvent, elles sont issues de sa connaissance sensible des "objets du monde"; la résolution suppose également que le sujet imagine un lien entre ces connaissances, ce lien pouvant être un "traitement à la main", c'est à dire d'un traitement dans les objets du monde, ou même un mode de résolution ne faisant pas appel à un traitement conscient (par exemple, repérer le caractère du milieu dans une chaîne de longueur impaire...): nous pouvons résumer la première ligne du tableau en disant que le sujet dispose d'un **S.R.T.** intuitif lui permettant de se représenter le problème, **S.R.T.** qui est influencé par son expérience sensible, qui est adapté à un traitement "dans les objets du monde" (en particulier l'élève dispose dans ce **S.R.T.** de *plans* au sens de [HOC 87]), mais pas à un traitement sur le dispositif informatique. La "*codification*" résulte d'un certain nombre de conventions "réductrices" du point de vue du sens, permettant d'associer aux connaissances données des entités informatiques; l'"*interprétation* " est l'opération inverse. Les données codifiées sont considérées pour leurs propriétés formelles, et non à partir d'un contexte intuitif; d'autre part, pour être adaptées à un traitement algorithmique, les données codifiées présentent des spécificités par rapport aux données intuitives (par exemple, une chaîne de caractères doit être indexée par un ordinal, une liste permet l'accès seulement à son premier élément et au reste, un booléen a seulement deux valeurs). Le raisonnement sur les données *codifiées* est donc d'une autre nature que le raisonnement sur les objets intuitifs, et par conséquent, pour produire une solution (un traitement pour un dispositif informatique donné) le sujet doit disposer d'un **S.R.T.** distinct du **S.R.T.** du problème intuitif, lui permettant de se représenter les données sous leur forme codifiée, et un traitement sur ces données.

Il y a problème de programmation si le sujet ne dispose pas d'un **S.R.T.** lui permettant de se représenter les données codifiées, et un traitement sur ces données. La résolution du problème est en fait la constitution de ce **S.R.T.**. Schématiquement, on peut distinguer deux cas de constitution de ce **S.R.T.**:

- le sujet ne peut s'appuyer sur des représentations mentales déjà constituées: dans ce cas, il doit déduire la solution en opérant des différenciations à partir du **S.R.T.** intuitif au fur et à mesure de la découverte des contraintes et spécificités du dispositif. Les conditions dans lesquelles s'opèrent ces différenciations sont certainement très diverses, dépendantes des connaissances antérieures des élèves, aussi bien que des caractéristiques du dispositif pour lequel les élèves programment. Par exemple, un dispositif de bas niveau, comme la *machine à registres* présentée dans [COHORS-FRESENBORG 87] (voir chapitre 2) comporte un ensemble de régles facilement énonçables et est susceptible de représentations matérielles; les fonctions de base (affectation, jeu d'instructions arithmétique...) devant être construites par le programmeur, les représentations mentales se construisent parallèlement. Au contraire, dans le cas d'un sous-ensemble d'un langage évolué (que nous avons rencontré au chapitre 2 avec la présentation de [SAMURCAY 85]), l'énoncé a priori des régles de formation serait sans intérêt, et, en tant que machine, ce sous-ensemble ne possède pas de représentation matérielle simple[57]. Dans ce cas, le sujet devra donc se constituer des représentations mentales opératoires à l'aide des exemples proposés par l'enseignant, à l'aide de

---

[56] J. ARSAC "L'informatique et le sens. Une gigantesque mutation culturelle ?" in Actes du colloque du CREIS histoire et épistémologie de l'informatique Mai 86.

[57] Des représentations matérielles sont parfois employées ; elles sont locales et métaphoriques, c'est-à-dire qu'elles illustrent une partie seulement du langage, et ne doivent pas être prises au pied de la lettre : par exemple, la métaphore de la pile d'assiettes pour la pile `LIFO` ne prend pas en compte le fait que dans `empiler (a)` la valeur de a n'est pas modifiée, alors qu'elle l'est dans `dépiler (a)`.

connaissances issues d'apprentissages dans d'autres disciplines, à l'aide des "effets en retour" du dispositif...[58].

- le **S.R.T.** adapté à un traitement algorithmique peut être constitué à l'aide d'éléments présents dans d'autres **S.R.T.**. Dans ce cas, le sujet doit opérer des relations et des différenciations entre **S.R.T.** Par exemple, dans un problème impliquant la représentation d'un graphe orienté (parcours sur un plan...), le sujet disposera d'un (de) **S.R.T.**(s) correspondant à un traitement sur un tableau numérique, et d'un (de) **S.R.T.**(s) comprenant des calculs sur les booléens. Il devra en quelque sorte "fédérer" ces **S.R.T.**, remettre en cause certains de leurs aspects trop liés aux contextes dans lesquels ils ont été construits, établir des relations avec le **S.R.T.** intuitif du problème.

En fait, si le problème s'adresse à des débutants, et s'il ne s'agit pas de la traduction directe d'un procédé de calcul vu dans les apprentissages antérieurs, le mode le plus courant sera celui décrit dans [HOC 87] et rapporté au chapitre 2 (§1.3): face à une tâche de programmation consistant à modéliser une situation qu'il connait, le sujet débutant ne trouve pas dans le **S.R.T.** correspondant au dispositif pour lequel il programme, un "*plan*" lui permettant d'imaginer une solution; il fait donc appel à des *plans* d'un **S.R.T.** servant dans la situation familière.

Ce mode de constitution éclaire les caractéristiques des **S.R.T.** dons nous avons parlé au §2.2. Le plus souvent, les **S.R.T.** familiers n'intègrent pas la notion de calcul sur les objets, et les traitements y sont liés aux objets: par exemple, dans un **S.R.T.** familier correspondant à un traitement sur les chaînes, les caractères sont repérés par des dispositions spatiales, et le traitement opère sur ces dispositions: ce n'est pas un calcul, contrairement au traitement des nombres dans le **S.R.T.** mis en oeuvre pour l'exponentiation. En construisant sa solution à partir d'un plan conçu pour la situation familière, le sujet **intègre dans son S.R.T. correspondant au dispositif informatique, des caractéristiques des représentations et traitements qui lui servent dans cette situation, et ceci d'autant plus largement que ces caractéristiques n'empêchent pas la résolution**: ainsi, par exemple, programmant le parcours d'une chaîne de caractères, le sujet pourra construire une itération fonctionnant effectivement tout en ayant une compréhension des éléments en jeu largement influencée par la situation familière: l'index pourra par exemple, ne pas être compris comme numérique, l'avertisseur d'itération `POUR i ← 1 JUSQU'A longueur(chaîne)` pouvant être compris comme donnant à la variable `i` son statut d'index parcourant la chaîne, et non comme organisant de façon précise l'évolution de `i` comme compteur de boucle.

3.2 Données et traitements dans les premiers apprentissages; discussion de quelques choix concernant les problèmes de programmation

Les hypothèses des paragraphes précédants, en particulier l'interdépendance que nous avons supposée au sein des **S.R.T.** constitués par le sujet en réponse à un problème de programmation, des éléments relatifs aux traitements, et des éléments relatifs aux données (§ 2.2), et le schéma de l'activité de programmation dans les premiers problèmes, fondé sur la différenciation entre le niveau intuitif où est posé le

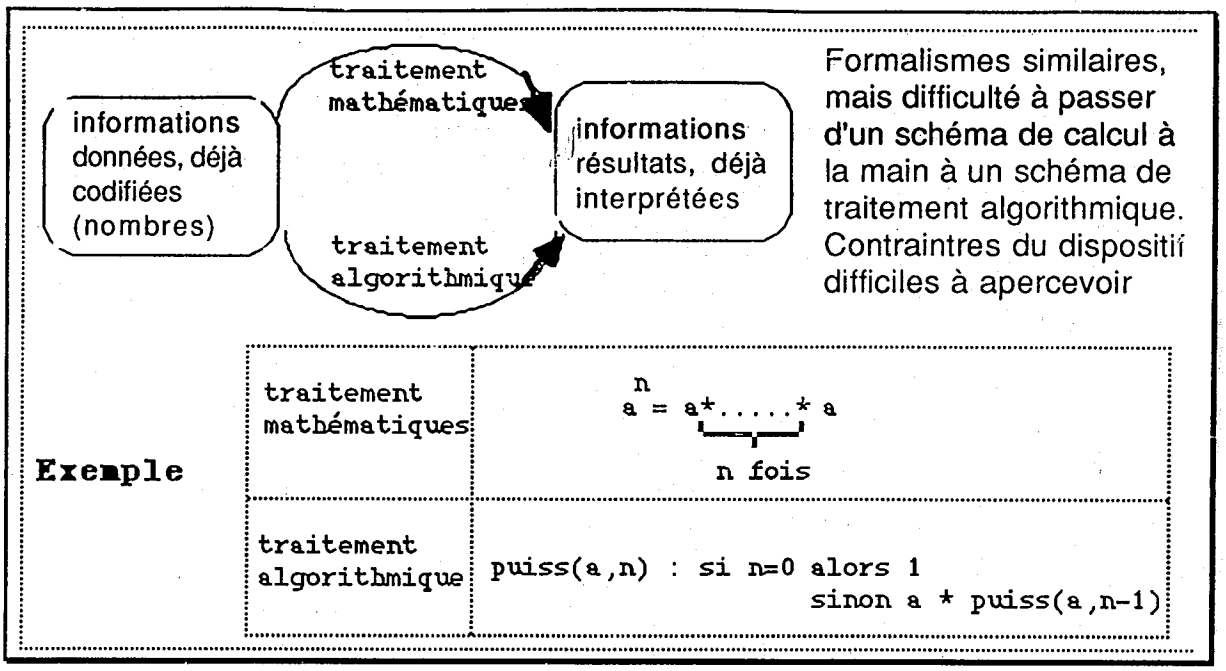
[58][ROGALSKI 87] (J. ROGALSKI «Savoirs et savoir-faire en informatique 1987 ») distingue deux voies possibles de constitution de nouvelles connaissances. Dans une première voie, "Le fonctionnement en lui-même des éléments d'un système inconnu peut permettre la constitution d'une signification des notions et des régles de ce système." J. ROGALSKI fait remarquer que, dans cette voie, le sujet "s'appuie souvent implicitement sur l'existence de systèmes de représentations et de traitements (...) suffisamment voisins pour introduire du sens au-delà des régles d'utilisation", et donc "Une autre voie est la construction de nouvelles connaissances à partir de celles de cadres préexistants avec des précurseurs(...)". Elle note que des précurseurs peuvent avoir un caractère facilitant pour les nouvelles représentations (précurseurs "producteurs"), ou au contraire un caractère inhibant (précurseurs "réducteurs"). Dans cet esprit, nous examinerons au chapitre 5 ce qu'il en est de ces précurseurs pour chacune des notions en jeu dans les apprentissages .

problème, et le niveau des données codifiées que suppose le traitement algorithmique (§ 3.1), nous conduisent à faire l'hypothèse que les problèmes proposés aux débutants seront d'autant plus intéressants que **le sujet pourra déduire du contexte intuitif un(des) plan(s) lui permettant d'amorcer une résolution, tout en imposant, à travers l'adaptation de ce(s) plan(s) pour le dispositif informatique des différenciations par rapport au contexte intuitif, allant dans le sens de l'abstraction des données (données considérées par leurs propriétés, multiplicité des représentations...) et de la reconnaissance du caractère général des traitements algorithmiques.** Pour mieux préciser le type de problème sur lequel nous ferons travailler les élèves dans la partie expérimentale de cette thèse, nous situons en premier lieu quelques "stratégies types" en les discutant par rapport à l'analyse qui précède.

3.2.1 Des problèmes où les premiers traitements portent sur des données "déjà codifiées"

Une stratégie peut consister à viser l'acquisition de capacités algorithmiques en laissant de côté la question de la codification des données, et donc à proposer des problèmes sur des données "déjà codifiées", c'est-à-dire des données numériques dans des problèmes qui se ramènent à la programmation d'algorithmes numériques. [SAMURCAY 85] et [COHORS-FRESENBORG 87], présentés au chapitre 2 s'intéressent à l'acquisition de capacités concernant les traitements sans considérer comme une variable la nature des objets sur lesquels portent ces traitements. Or les objets numériques sur lesquels interviennent les traitements dont ils étudient l'acquisition sont liés à un contexte de calcul numérique algorithmique et ont les particularités de ce contexte: R. SAMURCAY constate par exemple que, face à la programmation d'un algorithme d'exponentiation, les élèves qu'elle observe éprouvent des difficultés à passer d'une représentation de l'exponentiation issue des mathématiques à une représentation permettant la construction d'un traitement adapté au dispositif informatique.

Dans ce type de problème, en effet, le traitement en mathématiques et le traitement algorithmique sont isomorphes (le traitement par la machine peut en effet s'envisager comme une systématisation du "calcul à la main") mais supposent probablement chez le sujet des représentations différentes (représentation du calcul "déjà achevé" en mathématique, représentation du calcul "en train de se faire" en informatique). Plus précisément, en mathématiques, les procédures de calcul sont guidées par une définition statique et par conséquent ne dégagent pas de plans qui soient importables dans un **S.R.T.** informatique: par exemple, la définition de la puissance nième du nombre a comme produit de n nombres égaux à a ne constitue pas un plan de calcul bien que pour des valeurs particulières de a et n, elle permette une multiplicité de calculs. Pour surmonter ce type de difficulté, il faudrait sans doute que le sujet approfondisse à la fois sa conception du traitement mathématique et du traitement informatique, ce qui sort de notre propos. le schéma ci-dessous résume cette discussion.



Un autre écueil d'une stratégie où les algorithmes opèrent sur des données numériques, est que d'autres types interviennent: le type booléen comme type des conditions dans les instructions conditionnelles, les chaînes pour les entrées/sorties... Même si ces types jouent un rôle secondaire dans l'algorithme, et sont donc passés sous silence par l'enseignant, ils peuvent interagir fortement dans la compréhension qu'a le débutant du problème et de sa solution; nous avons montré au chapitre 1 (§ 4.5) que dans un problème de recherche du jour de la semaine où "tombe" le jour de Noël pour un millésime donné ([ARSAC 80]) la question du calcul du résultat sous forme d'une chaîne explicite (`Lundi, Mardi...`) peut, si elle n'est pas résolue, handicaper la recherche de l'algorithme de résolution par le débutant.

3.2.2 Plusieurs univers de programmation pour une méthode de construction d'algorithmes

Dans une autre direction, des auteurs comme [DUFOURD 88][59] ont proposé des "*univers de programmation*", comme situations *"stimulantes"* pour développer des capacités algorithmiques. Un univers de programmation est constitué, pour eux, d'un certains nombre de situations où un même type de donnée peut représenter les objets; ils proposent à titre d'exemple, un "*univers traditionnel*" constitué des booléens, des nombres et des chaînes, un univers graphique (la "*géométrie tortue*") et un "*univers de relation*" (les bases de données). Ils se proposent dans l'article, de montrer comment une même méthode de construction d'algorithmes peut opérer dans ces différents "univers". Par rapport à la stratégie précédente, cette direction a l'avantage de prendre en compte la dimension des "objets", mais, au vu de ce que nous avons discuté au chapitre 1, la conception des objets dans l'approche proposée par ces auteurs semble assez artificielle et séparée de la construction des algorithmes, puisqu'elle repose sur l'idée de situations "naturellement" liées à un type d'objet: il serait possible , selon les auteurs, d'obtenir a-priori une bonne structuration des données dans laquelle opérerait ensuite une méthode de construction d'algorithmes standard (la méthode MEDEE développée dans [DUCRIN 84]). Il n'y a donc pas pour eux, construction jouant de façon conjointe sur l'aspect structuration des données et sur l'aspect traitement, mais bien conception séparée de la structuration des données et de l'algorithme. En l'absence d'étude expérimentale, la façon dont se font les acquisitions des élèves dans ce cadre méthodologique reste hypothétique: les auteurs avancent

[59] [DUFOURD 88] G. et J.F. DUFOURD "Des univers et des outils variés pour commencer à programmer" in Actes du Premier Colloque Francophone de Didactique de l'Informatique Editions EPI

l'idée que les élèves pourraient s'approprier le langage lié aux objets par un fonctionnement "en mode bureau", puis pourraient ensuite appliquer la méthode MEDEE. Si les élèves débutants étaient capables d'appliquer directement une méthode où les concepts liés aux traitements sont d'emblée conçus comme indépendants des objets sur lesquels ils portent, cela voudrait dire que les **S.R.T.** qu'ils mettent en oeuvre comportent des éléments liés aux traitements conçus directement de façon indépendante des données, ce qui infirmerait nos hypothèses. Nous montrerons, à l'aide d'une étude expérimentale au chapitre 7, que ces hypothèses ont une validité et que, par conséquent il est raisonnable de mettre en doute l'efficacité de la méthode indiquée dans l'article.

3.2.3 L'approche de S.PAPERT

D'autres stratégies d'apprentissage font intervenir des environnement de programmation marqués par un type particulier; c'est le cas de la "*géométrie-tortue*" que [PAPERT 80][60] a introduit comme exemple de "*micro-monde*". Dans ce cas, il ne s'agit plus comme ci-dessus, de généraliser à plusieurs domaines de données une méthode de construction d'algorithme, mais bien de promouvoir un domaine particulier de données, en donnant comme raison qu'il serait le plus adapté aux acquisitions recherchées: «*L'arithmétique est une mauvaise introduction à la réflexion heuristique. En revanche, la géométrie Tortue en est une excellente. Grâce à ses qualités de syntonie (syntonie corporelle et syntonie du moi), le simple acte d'apprendre à dessiner à la tortue permet à l'enfant de se forger un modèle d'apprentissage fort éloigné du modèle dissocié*». Le domaine considéré paraît cependant très particulier: en tant que domaine mathématique, il s'agit d'une géométrie faisant largement appel aux mesures de distances et d'angles; en tant que domaine informatique, il fait intervenir un objet complexe constitué en fait comme un ensemble de variables globales non explicites: l'écran de dessin, la position , le cap, l'état du crayon... D'autre part, le domaine conceptuel où les acquisitions sont recherchées n'apparaît pas clairement: il s'agit d'acquérir un langage, donc en principe les acquisitions sont recherchées en informatique, mais il est souligné, en de nombreux points de l'ouvrage de S. PAPERT, que l'informatique n'est qu'un moyen. Des acquisitions en mathématiques seraient en fait recherchées, mais aussi des capacités transversales: acquisition de la pensée scientifique, heuristique...En revenant au domaine qui nous intéresse (les acquisitions en informatique) il se pourrait que des capacités sur un type d'objet aussi particulier soient assez peu transférables sur d'autres objets: [SAMURCAY ROUCHIER 90], suite à une étude expérimentale sur des élèves travaillant à l'écriture de procédures récursives, d'abord dans un environnement graphique, puis dans un environnement constitué de listes numériques, notent que «*le modèle relationnel qui a une valeur opératoire dans un cadre graphique, présente une grande fragilité. Dans un domaine nouveau (suites d'entiers) sa mise en oeuvre rencontre de difficultés importantes.*». En fait, le programmeur travaille, pour les problèmes les plus simples, dans un **S.R.T.** intuitif lié à son "schéma corporel", mais dès que les contraintes deviennent plus fortes, le **S.R.T.** qui serait nécessaire intègre des compétences en mathématique et/ou en informatique. J. HILLEL ([HILLEL 85]) a mené une série de recherches sur ce sujet. Dans le domaine des acquisitions en informatique, il ressort que, dans un environnement de géométrie tortue présentant des contraintes particulières, des élèves de 11 à 13 ans peuvent développer des capacités à construire un parcours de la tortue sous forme de procédures composées, et, dans une moindre mesure, à réutiliser pour d'autres tâches, une procédure écrite antérieurement. Les domaines où le sujet peut construire ces compétences à partir de la seule activité de programmation semblent donc en nombre plus limité que ce qu'avait.imaginé S. PAPERT.

3.3 Notre choix: des problèmes où les objets du monde réel et les données informatiques sont bien différenciés

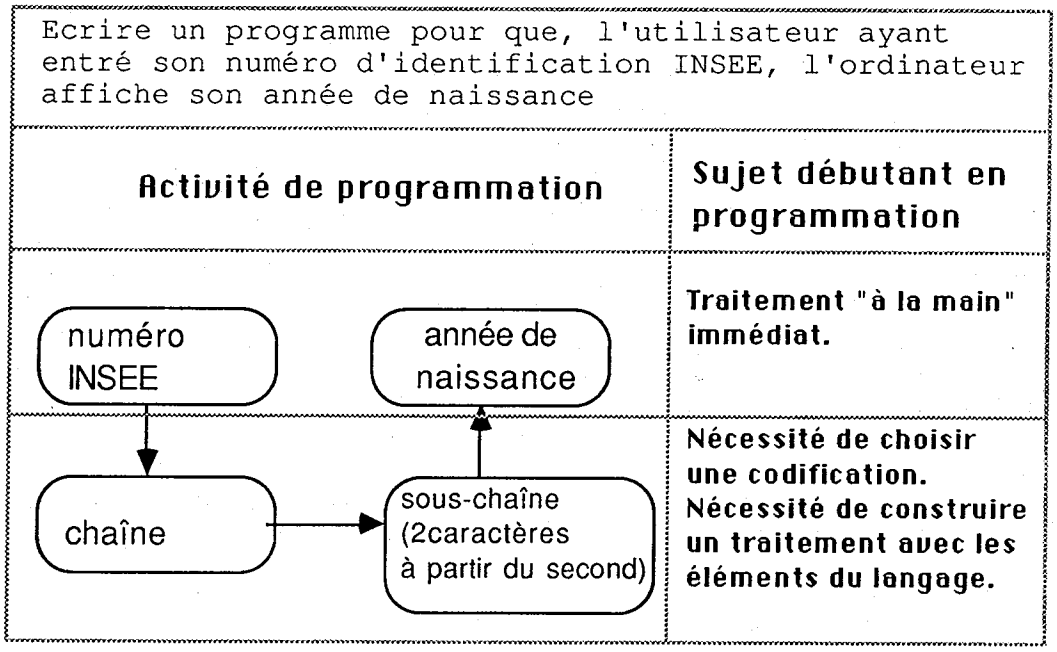
A la suite de l'analyse que nous avons développée en § 3.1, nous pensons

---

[60][PAPERT 80] : S. PAPERT MINDSTORMS Basic Books 1980 Traduit en français sous le titre de Jaillissement de l'esprit Flammarion 1981.

nécessaire de travailler sur des problèmes où la réalisation d'une tâche par un dispositif informatique impose de distinguer nettement le traitement formel sur les objets codifiés, du traitement dans les objets du monde. Il est important que, dans ces problèmes, la codification soit réellement repérable, et donc, d'une part, que le niveau où est posé le problème permette aux élèves de considérer de façon intuitive les objets en jeu et le lien entre données et résultats (et donc de disposer de représentations schématiques d'un traitement dans ces objets, c'est à dire de *plans* susceptibles de constituer un point de départ pour la résolution[61]) et d'autre part que la recherche d'une solution pour le dispositif informatique impose, quant à lui, de considérer ces objets sous forme de données informatiques et le traitement comme un "calcul" (direct ou algorithmique) sur ces données codifiées. De plus, les problèmes doivent faire intervenir de façon liée les concepts informatiques de base et au contraire, éviter de faire intervenir des éléments conceptuels (par exemple mathématiques) en dehors du champ de l'informatique; en effet les questions d'acquisition en informatique sont suffisamment complexes pour ne pas faire intervenir des difficultés d'analyse provenant d'acquisitions différenciées que pourraient avoir les sujets étudiés dans d'autres disciplines.

Nous pensons que les problèmes ainsi définis sont à même de faire rencontrer aux élèves les spécificités du dispositif lorsque la mise en oeuvre d'un *plan* (c'est à dire d'une représentation schématique d'une solution) importé du domaine familier imposera l'adaptation des **S.R.T.** de ce dispositif. Cette nécessaire adaptation entraînera des difficultés pour l'élève dans la construction de son programme, mais nous attendons que leur résolution conduise à l'intégration des contraintes et à une meilleure représentation du dispositif informatique. Nous précisons, au paragraphe suivant, la façon dont il est possible d'utiliser des types pré-définis des langages de programmation pour ces problèmes. Voici, au préalable, un exemple dans la figure suivante:



Le numéro d'identification INSEE est une chaîne de 13 chiffres attribuée en France à chaque résident par l'institut national de la statistique [CHAMBADAL][62]. Le

[61] Nous avons souligné que des problèmes revenant à la programmation d'algorithmes mathématiques ne permettent généralement pas à l'élève d'avoir recours à des plans de ce type.. Il en est de même de problèmes posés directement dans les objets informatique comme par exemple [ANDERSON 84] ANDERSON J.R. FARELL R. SAUERS R. 1984 Learning to program in LISP. *Cognitive Science 8*

[62] [CHAMBADAL 83] L.CHAMBADAL Calcul Pratique HACHETTE 1983

premier chiffre code le sexe (MASCULIN, FEMININ) de la personne, les deux chiffres suivants sont les deux derniers chiffres de l'année de naissance. Cet exemple peut paraître rudimentaire: nous verrons que sa programmation pose un réel problème à des élèves de seconde (partie III). Le traitement "à la main" est en effet immédiat: l'opérateur humain "sépare" les deux chiffres pertinents et les interprète aussitôt en tant qu'année: il n'a pas conscience du type d'entité (numérique ou chaîne de symboles) qu'il considère. Au contraire, pour programmer, il faut commencer par un choix de codification: si l'on choisit de codifier un numéro INSEE comme une donnée numérique, il faudra d'abord s'assurer qu'il existe un type d'entier dans le langage pouvant prendre une valeur à 13 chiffres décimaux, ce qui est rarement le cas. Si c'est le cas il faudra ensuite faire appel à des fonctions arithmétiques (division entière par 10^{10} puis reste modulo 100) Un autre choix est de considérer le numéro INSEE comme une chaîne de chiffres. Les fonctions sur les chaînes couramment présentes dans le langage (que nous présenterons au chapitre 5) permettent le calcul des deux derniers chiffres de l'année sous la forme d'une `sous-chaîne`, qu'il est possible de concaténer à 18 ou 19 pour obtenir l'année de naissance..

3.4 Quelle utilisation des types prédéfinis ?

Pour des débutants, la codification ne peut être séparée d'une représentation à l'aide des types prédéfinis dans le langage réellement utilisé, ce qui implique de raisonner sur cette représentation plutôt que sur des objets abstraits; le choix des types est réduit à ceux qui sont présents dans les langages de programmation les plus courants : types numériques, chaînes de caractères, et booléens. Pour chacun de ces types, nous nous proposons d'examiner la possibilité de construire des problèmes où le traitement sur les objets codifiés à l'aide de ces types se distingue suffisamment du traitement dans les objets du monde.

3.4.1 Les types numériques.

Ils sont conçus pour représenter dans un certain domaine de validité les entités mathématiques correspondantes (nombres entiers, nombres réels). La question du rapport entre les nombres tels que les élèves les conçoivent à la suite de l'enseignement de mathématiques qu'ils ont reçu, et la codification dans le dispositif informatique peut donc leur être posée à partir de problèmes où les entités numériques sortent du domaine de validité pour le langage utilisé : nombres entiers trop grands en valeur absolue, addition de nombres réels dont l'un est très petit par rapport à l'autre.[63]. Mais les réponses sont sans doute trop complexes dans le cadre de la première initiation et ce questionnement risque donc d'être peu productif.

En dehors de ce domaine, les types numériques sont assez peu adaptés pour le type de problème que nous envisageons. En effet, s'il est possible de codifier des données de nature diverse à l'aide de types numériques, le traitement a souvent, dans ce cas, recours à des fonctions mathématiques qui sont sans signification pour le problème posé. Dans l'exemple du paragraphe 3.3 , la représentation sous forme numérique du numéro INSEE conduirait à utiliser, pour isoler une partie du numéro, la division et le reste.

3.4.2 Les chaînes de caractères

Définition de la structure :

Nous avons indiqué ci-dessus que lors des premiers apprentissages, la

[63] Dans un langage de programmation, les nombres entiers ont nécessairement une valeur absolue limitée. Si un entier `A` positif est proche de cette limite, l'addition d'un autre entier `B` positif pourra soit entraîner une erreur, soit rendre un entier négatif. De même, la représentation des "nombres réels" en machine, sous forme d'une mantisse (nombre fractionnaire inférieur à 1) codée sous forme d'un nombre fixé de bits et d'un exposant variant dans un intervalle fini, permet de coder des nombres très petits et très grand, mais pour un nombre "grand" `A`, et un nombre "petit" non nul `x`, on peut très bien avoir `A + x = A`, ce qui est en contradiction avec l'arithmétique. Ce cas est étudié par exemple dans [MEYER BAUDOIN 80] (B.MEYER C.BAUDOIN. "Méthodes de programmation". Editions Eyrolles.) chapitre «introduction aux langages»

codification des éléments en jeu dans le problème ne peut se distinguer de la représentation dans le langage réellement utilisé, et que, en conséquence, il convient d'utiliser des représentations présentant un caractère suffisamment général ; c'est pourquoi nous limitons à 3 les fonctions sur les chaînes de caractères que nous utilisons avec les élèves (nous développerons ce point au chapitre 5, et nous présenterons la façon dont ces fonctions sont présentes dans les langages utilisés pour l'initiation):

- la fonction `sous-chaîne` a trois arguments : une chaîne, un ordinal (qui marque le rang du premier caractère de la chaîne résultat dans la chaîne argument), et un cardinal, représentant le nombre de caractères de la chaîne résultat.
- la fonction `longueur` dont l'argument est une chaîne, rend un nombre ; ce nombre peut être considéré comme un cardinal (nombre de caractères de la chaîne) ou comme un ordinal (rang du dernier caractère de la chaîne).
- l'opération de `concaténation` a deux arguments chaînes, et rend une chaîne .

Utilisation pour des problèmes :

Les chaînes de caractères ainsi définies permettent, dès l'initiation, de codifier des entités appartenant à des domaines sémantiques variés : des mots, des expressions , des listes linéaires d'objets représentés par des caractères ou des mots, et aussi des nombres considérés comme des chaînes de chiffres. La codification d'entités à l'aide de chaînes conduit à des traitements séquentiels typiques des traitements informatiques, et éloignés des traitements des mêmes entités par des opérateurs humains ; ainsi la recherche d'une occurrence d'une `sous-chaîne` dans une chaîne implique, dans le dispositif informatique, un traitement itératif et l'utilisation d'un index numérique, alors qu'un opérateur humain peut accéder directement au caractère cherché par appréhension globale de la chaîne.

Les traitements sur les chaînes utilisent la structure numérique sous les deux aspects, cardinaux et ordinaux : l'acquisition de la structure ordinale est intéressante à observer, en ce sens qu'elle est utile en informatique (calculs d'index sur les tableaux, par exemple), et qu'elle ne fait pas l'objet d'un enseignement au delà de la première année d'enseignement obligatoire (soit une dizaine d'année avant le début de l'enseignement d'informatique). Nous détaillerons ces propriétés de la structure de chaîne au chapitre 5. Dans la partie expérimentale de cette thèse, la structure de chaîne occupe une place importante puisque la partie II est une observation d'élèves en situation de résolution de problèmes impliquant la mise en oeuvre de cette structure, et la partie III un essai d'ingénierie didactique dans le même domaine.

3.4.3 Les booléens

Ils permettent de codifier des entités à deux valeurs, qu'il s'agisse d'objets ou de relations entre objets. Ils interviennent donc par exemple dans des problèmes concernant des dispositifs à deux états, et dans ce cas, la codification n'est pas immédiate pour le débutant, puisqu'elle est particulièrement réductrice. Organisés en tableau de booléens, ils interviennent dans des problèmes se ramenant à un parcours de graphe orienté (plan de ville, etc...); là aussi, la représentation spatiale familière, et le mode de raisonnement sur cette représentation s'éloignent de la représentation informatique et du traitement sur cette représentation, ce dernier traitement étant un calcul sur des booléens.

Par ailleurs, une conceptualisation adéquate des booléens paraît importante, puisque les traitements conditionnels (alternative, itération) font intervenir une condition qui est de type booléen. Jusqu'à ce que les élèves aient connaissance des booléens, le statut de cette condition, et des opérateurs propositionnels qui permettent de construire des conditions complexes reste flou, ce qui peut être gênant pour une utilisation systématique de ces structures (le passage d'une forme REPETER...JUSQU'A à la forme TANT QUE..., par exemple, conduit à considérer la négation d'une condition). Nous nous proposons de montrer dans la partie IV de cette thèse que des problèmes impliquant de représenter des objets du monde par des booléens contribuent à clarifier le statut des "conditions" auprès des élèves.

4. Conclusions du chapitre

Nous avons situé notre recherche au niveau des premiers apprentissages en informatique et indiqué que l'enjeu de ces apprentissages est l'accès des élèves aux concepts de base de l'informatique (variables, affectation, traitements alternatifs, traitements itératifs, typage des données...) leur permettant de dominer les premiers traitements dans des domaines de tâches et des langages variés, d'opérer les mises en relations, et les différenciations entre ces domaines et entre ces langages. Les concepts informatiques prenant leur sens d'abord dans des "problèmes de programmation" où ils interagissent, l'accès d'un débutant au niveau des concepts suppose qu'il se soit posé et qu'il ait résolu des problèmes divers, activité au cours de laquelle il se constitue un ensemble des représentations mentales que [HOC 77] désigne par Systèmes de Représentation et de Traitement (**S.R.T.**) . Pour nous, un **S.R.T.** est la structure mentale locale qui permet à un sujet, face à une tâche de programmation donnée, de construire une solution et/ou de vérifier sa validité. Le niveau des concepts est celui où le sujet, disposant de **S.R.T.** relatifs à des tâches diverses, est capable de les mettre en relation et de les différencier, de façon à opérer des transferts lorsque se présente une tâche nouvelle, et à pouvoir raisonner sur les concepts indépendamment des domaines de tâches.

Nous avons distingué des plan(s) relatif(s) aux objets.et des plan(s) relatif(s) aux traitements, s'organisant en sous-systèmes au sein d'un **S.R.T.**, de façon à permettre la prise en compte des spécificités des conceptions du sujet relatives aux objets, dans la résolution d'un problème donné à un moment donné de ses acquisitions, et des interactions entre les conceptions des objets et les conceptions des traitements. Dans le type d'initiation que nous envisageons, le mode le plus courant de constitution de **S.R.T.** à travers la résolution de problèmes, est l'importation de *plans* issus de **S.R.T.** adaptés à des traitements dans des domaines familiers (extra-informatiques). Il conduit à ce que les plans relatifs aux objets conservent dans le **S.R.T.** construit pour la recherche d'une solution informatique, des propriétés du contexte intuitif, et à ce que les plans relatifs aux traitements soient conçus de façon dépendante des objets sur lesquels ils portent.

Pour se constituer des **S.R.T.** adaptés, le sujet débutant devra donc être confronté à des problèmes de programmation qui, tout en permettant l'importation de plans à titre heuristique, lui imposent d'opérer des différenciations par rapport à ses conceptions des objets et traitements intuitifs. En particulier, il devra progressivement concevoir les objets informatiques comme *calculables*, et donc progresser vers l'utilisation de définitions abstraites, intégrer le caractère général des traitements...Nous en avons déduit que les énoncés doivent avoir les spécificités suivantes:

- le niveau où est posé le problème permet que les élèves considèrent de façon intuitive les objets en jeu et le lien entre données et résultats; plus précisément, le **S.R.T.** mis en oeuvre pour ce niveau permet à l'élève de se représenter l'état initial et l'état final, et contient des représentations schématiques de traitements.
- le niveau du traitement par le dispositif informatique impose, quant à lui, une "codification" de ces objets sous forme de données informatiques et la production d'une solution comme "calcul" sur ces données codifiées;
- les problèmes font interagir des concepts informatiques de base (objets, traitements) et au contraire, évitent de faire intervenir des éléments conceptuels (par exemple mathématiques) en dehors du champ de l'informatique.

Nous avons montré que les types numériques sont assez peu adaptés pour ce type de problème. Avec les chaînes de caractères et les booléens, il doit, au contraire, être possible de construire des problèmes utilisant ces types pour les codifications. Partant du cadre général tracé dans ce chapitre, le chapitre suivant précisera les hypothèses et le cadre spécifique pour l'étude expérimentale des parties II, III et IV.

Chapitre 4 Choix spécifiques pour l'expérimentation

hypothèses sur le fonctionnement cognitif et les problèmes
questionnement sur la situation pédagogique
l'option informatique comme terrain
planification de l'observation
méthodologie expérimentale

Au chapitre précédent nous avons indiqué dans quel cadre général s'insère cette thèse: choix du public observé, fonctionnement cognitif du sujet et problèmes de programmation en alphabétisation informatique. A l'intérieur de ce cadre général, le présent chapitre délimite le domaine d'investigation de notre thèse: hypothèses spécifiques sur le fonctionnement cognitif et les problèmes, questions concernant une situation pédagogique. La validation de ces hypothèses, et des réponses à ces questions supposent une étude expérimentale: nous décrivons donc également dans ce chapitre le cadre scolaire constituant le terrain pour cette étude expérimentale, nous discutons l'organisation dans le temps, et la méthodologie d'observation que nous adoptons pour cette étude expérimentale.

1. Hypothèses et questionnement

A la suite des conclusions du chapitre précédent, nous posons, pour le travail expérimental que constitue la suite de cette thèse, deux séries d'hypothèses et un questionnement:

1.1 Hypothèses sur les difficultés des élèves, et la constitution des S.R.T.

- **L'ampleur des difficultés** rencontrées chez une fraction notable des débutants (voir ci-dessous § 2.3) nous conduit à faire l'hypothèse que les concepts de base de l'informatique, contrairement à ce qui a été souvent considéré, ne sont pas disponibles de façon spontanée pour les débutants. Chez beaucoup d'élèves, la construction de représentations mentales adéquates du dispositif, s'organisant en **S.R.T.** différenciés et mis en relation, est un processus à la fois lent et nécessaire pour l'émergence des concepts. Ce processus implique un travail de mise en évidence et de remise en cause de **S.R.T.** non adaptés aux traitements informatiques.
- Dans les **S.R.T.**, nous distinguons les **plans relatifs aux objets** interagissant avec des **plans relatifs aux traitements**; les **plans relatifs aux objets** résultent de l'intériorisation par le sujet des propriétés des objets dans le domaine de tâche considéré, celles-ci étant influencées, à travers la résolution des problèmes, par les propriétés d'objets dans des contextes intuitifs (par exemple, la compréhension des propriétés des objets du type "chaîne de caractères" est influencée est influencée par la compréhension d'actions familières sur des textes): dans un **S.R.T.** construit pour une tâche de programmation dans un langage impératif (voir chapitre suivant), le plan relatif à un type d'objet est constitué des significations données par le sujet aux variables représentatives des objets de ce type, aux fonctions sur ces variables et aux éléments du langage liés à ces variables, en premier lieu **l'affectation**, et **les écritures fonctionnelles.** Les difficultés d'intégration de ces structures relatives aux objets, la spécificité des relations entre conceptions relatives aux objets et conceptions relatives aux traitements dans un **S.R.T.** donné, nous semblent avoir été négligés jusqu'ici (voir exemples chapitre 3 §2). Nous faisons donc l'hypothèse que la constitution de plans relatifs aux objets permettant la construction d'un traitement pour le dispositif est une étape obligée de la constitution de **S.R.T.** adaptés.
- L'interaction, dans la résolution d'un problème d'informatique, de la structuration des données et de la structure algorithmique, nous conduit à faire l'hypothèse que,

face à un problème mettant en jeu un traitement complexe (par exemple des alternatives imbriquées, ou une itération), **les plans relatifs aux traitements se construisent dans le S.R.T. du sujet en relation étroite avec les plans relatifs aux objets**, et donc que, tant que le sujet raisonne au niveau du domaine de tâches et non au niveau des concepts, **il ne peut envisager la création d'un traitement indépendamment de sa conception des objets informatiques en jeu**. Nous complétons donc l'hypothèse ci-dessus: la constitution de **S.R.T.** adaptés suppose de considérer et de faire évoluer de façon conjointe les plans relatifs aux objets et les plans relatifs aux traitements.

Conséquences de ces hypothèses:

Il nous faut vérifier en premier lieu que la codification d'objets intuitifs par des entités informatiques et les calculs simples (c'est-à-dire ne mettant pas en oeuvre des structures algorithmiques autres que la séquentialité) correspondant à des manipulations sur les objets intuitifs, ne vont pas de soi pour des débutants, qu'ils leur imposent un réel travail de construction de représentations mentales adéquates, un réel travail d'abstraction. Plus précisément, nous tenterons, au chapitre **5**, en ce qui concerne les chaînes de caractères, et au chapitre **10** en ce qui concerne les booléens, de cerner, à partir des éléments du langage, et de ce que nous pouvons prévoir du fonctionnement cognitif des élèves, des *difficultés*, traduction concrète de conceptions relatives aux objets inadéquates chez ces élèves. Puis, dans une étude expérimentale, nous examinerons comment ces *difficultés* peuvent ou non être repérées chez des élèves (chapitres **6** et **11**), et nous tenterons de mieux comprendre comment elles interviennent dans les conduites. Le terme de "*difficulté*" est volontairement vague. En effet, dans une phase d'alphabétisation qui s'étale sur peu de temps (rarement plus d'une année) et occupe relativement peu de séances, nous ne pouvons savoir a priori si ces conceptions erronées sont seulement transitoires chez le sujet, ou vont se transformer en *obstacles*. Nous pensons a priori, que les *difficultés*, en tant que constitutives d'un **S.R.T.** lié à un domaine de tâches donné, ont un ancrage local, et sont relativement résistantes, en ce sens qu'elles se retrouvent dans les conduites du sujet concerné, face à différents problèmes liés à ce domaine, et persistent malgré le "feed-back" de la machine ou de l'enseignant, tant que le sujet n'a pas reconsidéré la conception que la difficulté traduit. Nous soulignons que, constituant la première phase d'un enseignement d'informatique, l'alphabétisation ne saurait "hériter" de *conceptions* installées par des enseignements antérieurs, et qui feraient *obstacle* aux nouvelles connaissances, comme c'est souvent le cas en mathématiques. [64].

Puis nous devrons valider notre hypothèse concernant les conséquences pour la création des programmes par un sujet, de l'interaction, au sein de ses **S.R.T.** des plans relatifs aux objets et des plans relatifs aux traitements. Il nous faudra donc faire intervenir des problèmes impliquant des traitements plus élaborés que la simple séquentialité (itération, alternatives imbriquées), et observer comment les difficultés définies ci-dessus, vont se retrouver et interagir avec les difficultés propres au traitement complexe choisi, quelles nouvelles difficultés vont apparaître, liées aux particularités des représentations spontanées à partir d'une conception intuitive du traitement considéré.

1.2 Hypothèse sur les problèmes susceptibles de mettre à jour les difficultés et de faire évoluer les S.R.T.

Les éléments de représentation erronés dans les **S.R.T.**, que nous avons postulé ci-dessus, ont un domaine de validité non vide, et de nombreux énoncés peuvent en fait se situer dans ce domaine de validité, ou, plus simplement, ne pas les solliciter. Notre but est par conséquent de construire et de tester des énoncés où ces représentations peuvent être conçues par l'élève comme adéquates, mais se révèlent d'une manière ou d'une autre, entrer en contradiction avec la logique du dispositif informatique. Nous avons montré au chapitre précédent que ces énoncés ont les spécificités suivantes:

- le niveau où est posé le problème permet que les élèves considèrent de façon

[64] Pour une discussion sur les notions de conception et d'obstacle, voir [ARTIGUES 91]

intuitive les objets en jeu et le lien entre données et résultats;

- le niveau du traitement par le dispositif informatique impose, quant à lui, une **"codification" de ces objets sous forme de données informatiques et la production d'une solution comme "calcul" sur ces données codifiées;**
- les problèmes font interagir des concepts informatiques de base et au contraire, évitent de faire intervenir des éléments conceptuels (par exemple mathématiques) en dehors du champ de l'informatique.

L'hypothèse porte donc sur:

- la **possibilité de construire de tels problèmes** à l'aide des types de base des langages couramment utilisés pour l'initiation,
- l'utilité de ces problèmes pour la **mise en évidence du fonctionnement cognitif** des élèves,
- la possibilité de les organiser en "**progression**" visant à la mise en cause et à l'amélioration des **S.R.T.**

1.3 Questions posées par la situation pédagogique: le cas du travail dirigé en présence de l'ordinateur

La situation pédagogique, c'est-à-dire l'ensemble des interactions entre l'(les) élève(s), le maître, le problème de programmation, le dispositif informatique et les concepts, constitue un système complexe où le choix du problème (présenté ci-dessus) n'est qu'une variable parmi d'autres. En particulier, l'intervention du professeur ne saurait être évacuée de l'analyse, même si elle ne peut être caractérisée de façon immédiate dans la situation de "Travail Dirigé" qui domine l'alphabétisation: nous ne pensons pas que l'élève puisse seul, face au problème et au dispositif, construire de façon économique des représentations adaptées, et que le travail du professeur se réduise à une institutionnalisation des concepts a posteriori. L'absence de tradition en enseignement de l'informatique, particulièrement au niveau de l'alphabétisation et le nombre réduit d'études didactiques faisant de façon précise l'analyse des situations pédagogiques dans ce domaine, nous paraît rendre difficile la construction a priori de situations pédagogiques[65]. Plus que d'hypothèses, le travail sur ces situations nous paraît donc devoir, dans l'état actuel de la recherche, faire l'objet d'un questionnement. Les **situations de résolution par petits groupes, encadrées par l'enseignant, l'ordinateur étant à disposition pour des essais de compilation et/ou d'exécution** nous intéressent prioritairement. En effet, les pratiques d'alphabétisation informatique auxquelles nous avons participé avant cette thèse, nous conduisent à considérer ces situations comme les plus productives: les situations collectives, d'enseignement magistral, ou de résolution collective se heurtent en effet à l'hétérogénéité des représentations et des aptitudes qu'elles entraînent. Ceci constitue évidemment une impression subjective et non un résultat, mais, comme choix d'un *angle d'attaque*, la situation qui nous paraît la plus riche est aussi celle pour laquelle nous pouvons le plus facilement nous poser des questions:

- **la présence du dispositif informatique** est une spécificité de cette situation de résolution: comment les contraintes d'expression et les effets en retour par lesquels il se manifeste (résultats d'exécution, messages d'erreurs à la compilation et à l'exécution) interviennent-ils dans la construction des représentations des élèves?
- contrairement à d'autres situations de résolution, **l'enseignant** n'a pas un rôle d'organisateur de la communication. Pourtant il intervient auprès des groupes, sollicite, conseille... Dans les pratiques que nous connaissons, ces interventions sont

---

[65] L'existence d'un ensemble d'ouvrages de niveau universitaire présentant les savoirs en informatique nous a permis de repérer comme pertinente l'analyse de la forme sous laquelle le savoir est considéré (forme recontextualisée, forme décontextualisée) élaborée par [BROUSSEAU 86] pour la didactique des mathématiques. Le manque de référence que nous signalons rendrait hasardeux l'importation directe de la modélisation des situations pédagogiques qui est un autre aspect important du travail de G. BROUSSEAU. Cependant, à travers l'observation-questionnement que nous projetons, nous examinerons la pertinence, dans le domaine qui nous intéresse, de certains concepts crées pour cette modélisation (Chapitre 9)

très dépendantes des choix de résolution et de l'état d'avancement de chaque groupe, très liées également à l'interprétation qu'il fait des productions des élèves. Selon quels axes peut-on classer ces interventions ? Sur quels éléments l'enseignant peut-il s'appuyer pour interpréter les productions des élèves ?
- comment se fait la **communication** dans le groupe, et avec l'enseignant ?

2. Le terrain: l'option informatique des lycées; classe de Seconde et de Première

Nous avons expliqué au chapitre précédent pourquoi, dans l'état actuel de la didactique de l'informatique, la recherche doit partir de l'observation d'un public hétérogène sans finalité professionnelle. Dans cette perspective, l'option informatique des lycées (qui s'adresse à des élèves des sections classiques et de certaines sections techniques de l'enseignement long post-obligatoire) constitue un terrain adapté pour plusieurs raisons. D'une part, ayant dépassé le stade expérimental, les objectifs de cet enseignement ont été largement débattus et clarifiés, et font l'objet d'une publication officielle. D'autre part, la classe de Seconde est maintenant ouverte à la grande majorité d'une classe d'age, et bien qu'optionnel, l'enseignement s'adresse à tous: l'hétérogénéité qui doit en résulter est effective en classe de Seconde, sauf politique restrictive des établissements[66] . Les quelques éléments qui suivent visent à préciser certains aspects de l'option informatique de façon à délimiter la part de contingences qu'apporte le choix de ce terrain à l'étude expérimentale rapportée dans les chapitres suivants.

2.1 Programmes et pédagogies de l'option informatique

Les modalités d'organisation et les programmes de l'option sont des arrêtés ministériels publiés au bulletin officiel de l'Education Nationale et regroupés dans une brochure éditée par le CNDP [CNDP 87][67]. L'horaire est de 2h30, dont une heure de cours (plein effectif, soit en principe jusqu'à 40 élèves) et 1h30 de Travaux Pratiques (en demi-classe, avec présence des ordinateurs). Les objectifs généraux précisent qu'il s'agit d'un «*enseignement à caractère général*», donc non «*orienté vers un type d'application ou de profession*». Il s'organise en «*apport de connaissances techniques*», «*apprentissage de méthodes de travail*», et «*prise de conscience des enjeux économiques, sociaux et culturels*».

Le programme de la classe de Seconde recouvre les notions de base de l'informatique (données simples et leurs représentations internes et externes, traitements séquentiels, traitement conditionnel, traitement itératif). Les notions de tableau, de fichiers de données, de procédures et de fonctions sont au programme de Première.

Les directions pédagogiques officielles sont fortement influencées par le travail de J.ARSAC: méthode de résolution en 5 points («*Enoncé d'un problème; Découverte et expression d'un procédé de résolution; Codage dans un langage de programmation; Validation du programme ainsi obtenu; Utilisation effective pour l'application retenue*»), méthode d'enseignement faisant intervenir, comme point de départ, le développement d'«*exemples typiques*» par le professeur. Cependant, il nous semble que ces directions ne doivent pas être conçues comme un point d'achèvement normatif; le débat sur les méthodes et les contenus est largement ouvert, comme le montrent les articles parus dans la revue **Informatiques** (par exemple [PAIR 88], et les nombreuses contributions des professeurs de l'option).

---

[66] Devant l'afflux de candidatures pour l'option, certains établissement sont tentés d'opérer une sélection des élèves sur la base des acquis scolaires antérieurs, ce qui est en contradiction avec l'esprit et les textes de l'option. Bien qu'étant moins générale que ce que laisse supposer [M.E.N. 91], cette politique se rencontre. Nous nous sommes assurés que les classes où nous avons mené les observations n'ont pas fait l'objet d'une telle sélection.

[67] [CNDP 87] OPTION INFORMATIQUE Classes de seconde, première et terminale.CNDP 1987

2.3 Problèmes didactiques de l'option.

Nous avons dit que l'hétérogénéité est effective en classe de Seconde, sauf politique restrictive des établissement. Par contre selon [BARON 89][68], à partir de la classe de Première les élèves qui persistent dans cet enseignement sont en grande majorité des élèves de sexe masculin des classes scientifiques. Le taux important de non-continuation de l'option après la classe de Seconde, particulièrement marqué chez les élèves non-scientifiques et les filles, est évidemment un des problèmes importants, puisqu'il remet en cause l'option dans sa finalité d'enseignement ouvert à tous. Les professeurs de l'option sont conscients de ce problème: certains ([BENETOLLO 90][69]) insistent sur une prise en compte de l'hétérogénéité des élèves à partir de la situation pédagogique de Travaux Pratiques; d'autres ([AUTHIER, WAITER 89][70]) analysent les difficultés des élèves comme «liées au passage d'une situation concrète à sa représentation en termes d'objets abstraits». Ces analyses vont dans le sens des questions que nous nous posons au départ de cette thèse, mais, selon nous, la portée des débats est quelque peu limitée par le manque d'études réellement didactiques à support expérimental dans le domaine des premiers apprentissages en informatique.

2.4 Le cas des classes observées

Nous avons pu mener nos observation dans des classes où enseignaient des professeurs largement impliqués dans la réflexion pédagogique autour de l'option (un enseignant est le coordinateur académique, une autre encadre des formations pour les enseignants de l'option). Ceci explique que le programme et les choix pédagogiques adoptés dans ces classes sont en conformité avec les instructions officielles: les notions abordées en classe de Seconde, et l'hétérogénéité constatée se sont donc trouvées tout à fait adaptées à l'observation que nous projetions, au moins pour les parties concernant les chaînes de caractères (CF ci-dessous §1.1, 1.2 et 1.3). La répartition des notions entre la Seconde et la Première, ainsi que des contingences liées au langage utilisé dans les classes de Seconde auxquelles nous avions accès, nous ont conduit à mener l'observation liée au type booléen (CF ci-dessus §1.4) en classe de Première, au prix d'une moins grande hétérogénéité, dont il est important de tenir compte pour l'exploitation des résultats.

Comme toutes les classes de l'option, les classes observées utilisaient un langage "évolué" (symbolique). Nous avons vu au chapitre 3 (§ 3.1) que ceci n'est pas indifférent du point de vue de la constitution des **S.R.T.**: le sujet programmant en langage évolué ne peut en effet s'appuyer sur des représentations matérielles entièrement adéquates, ni sur des règles de fonctionnement totalement explicites, et doit donc utiliser les exemples proposés par l'enseignant, des connaissances issues d'apprentissages dans d'autres disciplines, les "effets en retour" du dispositif... Comme dans beaucoup de classes de l'option, le langage était impératif (BASIC ou PASCAL suivant les classes), l'important pour nous étant que le langage permette d'utiliser commodément les types de base (chaînes et booléens). Nous discuterons au chapitre suivant l'influence du choix d'un langage impératif, et la disponibilité des types de base dans les différents langages.

Nous n'avons pas particulièrement recherché des classes à faible effectif, mais il s'est trouvé que les classes où nous avons travaillé comptaient entre 13 et 15 élèves. Ce faible effectif s'est trouvé compatible avec les premières observations de type clinique (voir ci-dessous). Nous avons dû ensuite vérifier leur généralité par une épreuve portant sur plusieurs classes. La classe où nous avons expérimenté une progression d'exercices (Cf 1.3 Essai d'"ingénierie didactique") disposait de la

---

[68][BARON 89] G.L. BARON L'option informatique à la rentrée de 1988/89 in INFORMATIQUES n°6 2nd trimestre 89.

[69][BENETOLLO 90] R. BENETOLLO Hétérogénéité et réussite dans les classes d'option informatique des lycées Actes du 2ème colloque francophone sur la didactique de l'informatique. Septembre 90.

[70][AUTHIER, WAITER 89] A. AUTHIER, N. WAITER Modèles, objets et pédagogies de l'option informatique in INFORMATIQUES n°6 2nd trimestre 89.

structure "Travaux Pratiques" (ordinateur pour chaque groupe de 2 élèves) sur la totalité des 2h30 hebdomadaires, ce qui a constitué un élément facilitant pour cette expérimentation (l'heure hebdomadaire de cours n'aurait pas eu d'intérêt dans cette expérimentation compte tenu des choix que nous avons indiqué en 1.3).

3. Planification de la recherche

Les hypothèses à tester, et le questionnement concernant la situation pédagogique de travail dirigé (§1) conduisent à deux types types de travaux énoncés ci-dessous en 3.1 et 3.2.

3.1 Observer de façon ponctuelle à des moments opportuns, des élèves ayant constitué, dans un cadre pédagogique donné, des ébauches de représentations

Cette observation a pour but de vérifier la pertinence des hypothèses concernant le fonctionnement cognitif et les *difficultés* qui en résultent (ci-dessus 1.1). Pour cette observation, le *type de problème* que nous avons défini en 1.2 est évidemment mis à contribution. La discussion du chapitre précédent concernant l'utilisation des types prédéfinis, et l'ordre dans lequel ces types sont abordés dans les classes conduit à centrer une première observation sur les problèmes utilisant les chaînes de caractères et les types associés (ordinaux et cardinaux), puis à l'élargir aux booléens. Cette observation comporte donc trois temps:

3.1.1 Observation de "*difficultés*" liées aux structures relative aux objets dans le cas de tâches de programmation sur les chaînes de caractères

Nous devons faire des hypothèses sur les *difficultés* spécifiques, choisir une classe et le moment adapté (il faudra que les élèves aient commencé des exercices sur les chaînes, mais pas nécessairement les traitements complexes), prévoir une épreuve et des entretiens à partir d'une série limitée d'exercices (cette observation est rapportée en partie **II** chapitre **6**). Les exercices, conformes aux spécifications énoncées en 1.2 devront faire intervenir la structure de chaîne, l'affectation et la séquentialité, que nous pensons constitutifs des plans relatifs aux objets, et exclure les "traitements complexes" (itération, alternatives), de façon à séparer les *difficultés* observées de celle qui interviennent dans le cas d'un traitement complexe..

3.1.2 Observation de l'interaction dans un S.R.T., des plans relatifs aux objets, et des plans relatifs aux traitements dans le cas d'un traitement sur une chaîne de caractères

Il nous faudra ici choisir un type de traitement algorithmique interagissant avec la structure de chaîne, faire des hypothèses sur la façon dont le fonctionnement cognitif des élèves face à ce type de traitements peut se traduire en conduites, construire une épreuve et faire passer des entretiens. (observation rapportée en partie **II** chapitre **7**)

3.1.4 Par la suite, il nous faudra aussi montrer que les résultats observés dans ce cadre ne sont pas limités aux chaînes de caractères

Au chapitre précédent, nous avons examiné les possibilités d'exploitation des codifications utilisant les booléens: codification d'objets à deux états à l'aide de variables booléennes, codification de relations à l'intérieur d'ensembles d'objets à l'aide de tableaux de booléens. Il parait donc judicieux de prolonger le travail sur les chaînes par une observation d'élèves en situation de résolution de problèmes impliquant l'emploi de variables booléennes, les objectifs de cette observation étant, comme pour les chaînes, de mettre en évidence des plans relatifs aux objets booléens, les difficultés crées par l'existence de conceptions erronées, leurs spécificités, leur interaction avec les plans relatifs aux traitements, leur apport aux apprentissages généraux en informatique...

3.2 Tester la possibilité d'organiser la *classe de problèmes* que nous avons définie ci-dessus, en *progression* visant à faire évoluer les S.R.T. des élèves, et voir comment répondre aux questions concernant la situation pédagogique

Ceci conduit à

3.2.1 un travail d'"ingénierie didactique"

Après les observations envisagées en 3.1.1 et 3.1.2, et en fonction de la compréhension du fonctionnement cognitif des élèves qu'elles nous auront apporté, il nous sera possible, en faisant le choix d'un champ de connaissances bien délimité, de construire un ensemble de problèmes de programmation, organisé en **progression** et visant à des acquisitions dans ce champ. Ce travail de construction nous conduira à analyser de façon plus précise que dans ce chapitre, les choix possibles concernant les énoncés, et à classer ces énoncés suivant une typologie relative aux aspects du fonctionnement cognitif des élèves sur lesquels il nous semble possible d'intervenir. (partie **III** chapitre **9**)

3.2.2 une observation d'élèves à partir de la progression ainsi construite

Cette observation doit nous permettre d'une part d'examiner les points forts et les points faibles de la progression et donc de l'analyse qui y a conduit, et d'autre part, de permettre un examen des questions posées en 1.3 concernant la situation pédagogique de travail dirigé par petits groupes; il nous faudra préciser le choix des élèves observés, le choix du moment dans les apprentissages où s'insère cette progressions, le choix la procédure d'observation. (partie **III** chapitre **10**).

4. Méthodologie

4.1 Les choix méthodologiques possibles

Une recherche en didactique suppose en premier lieu que les hypothèses avancées, les questions posées, le soient à partir d'une étude du domaine scientifique concerné, et des conditions psychologiques de l'apprentissage des notions de ce domaine. C'est ce que nous avons tenté de faire jusqu'à ce point de notre thèse. La recherche suppose également que ces hypothèses et questions soient confrontées à la réalité, sous la forme d'une étude expérimentale: seule cette confrontation permet de décider de la validité de ces hypothèses, de savoir si les questions ont été bien posées, quelles hypothèses supplémentaires on peut en attendre. Le cadre même des questions que nous nous posons conduit de façon directe au choix du terrain expérimental: nous avons montré au chapitre précédent que le terrain des premiers apprentissages est, dans l'état actuel de la recherche le seul possible, et dans ce chapitre, que l'option informatique des lycées convient tout à fait. Il nous reste à préciser le type d'étude expérimentale qui convient à nos hypothèses et questions: schématiquement le chercheur peut mener deux types d'études:

- une étude clinique: celle-ci consiste à repérer des conduites, à les classer, mais également à approfondir par une étude spécifique auprès d'élèves choisis, les raisons des ces conduites, de façon à les relier aux hypothèses concernant le fonctionnement cognitif.
- une étude quantitative: elle consiste à recueillir des données sur un nombre important d'élèves, concernant des conduites reliées aux hypothèses cognitives , de façon à en montrer la généricité, à établir des relations entre facteurs.

L'étude clinique convient particulièrement lorsque la définition des hypothèses, le lien entre les aspects du fonctionnement cognitif sur lesquels elle porte et les conduites, sont problématiques. Il faut souligner que dans ce cas, le processus d'élaboration des hypothèses et la confrontation à la réalité se conduisent dialectiquement: les premières conduites observées conduisent à approfondir le cadre d'analyse des processus cognitifs et donc à préciser les hypothèses, ce qui, à son tour, permet de mieux centrer l'observation. L'étude clinique, ne prétendant pas à la vérité statistique, peut se mener sur relativement peu de sujets, à condition de repérer les particularités

(apprentissages antérieurs...) de ces sujets, et d'éviter des cas trop particuliers pouvant interférer entre le fonctionnement cognitif et les conduites. Dans le cas où la problématique est clairement établie, et les conduites clairement reliées aux processus cognitifs, l'étude quantitative permet d'établir des vérités de nature statistique.

Le type d'activité des élèves, support de l'observation, compatible avec le cadre cognitif dans lequel nous nous situons est la résolution de problèmes de programmation. Ceci suppose donc que l'observation qu'elle soit de nature clinique ou quantitative, parte d'énoncés proposés aux élèves. (nous désignerons dans la suite de cette thèse, par *épreuve* , un ensemble d'énoncés proposés aux élèves). Nous verrons que, selon les contenus visés, la totalité de la tâche d'élaboration d'une solution peut être à la charge de l'élève, ou, au contraire, certains éléments de résolution (en particulier concernant le codage de l'information) sont précisés dans l'énoncé, par exemple sous la forme d'une amorce de programme, et la tâche de l'élève est de compléter ce programme à partir de sa compréhension des éléments déjà présents. Il est clair qu'une étude de type quantitatif ne pourra laisser trop d'éléments ouverts si l'on veut pouvoir procéder à un dépouillement statistique. D'autre part, une étude de type quantitatif peut difficilement prendre en compte des réponses d'élèves résultant d'une activité de programmation allant jusqu'aux essais de programmation sur le dispositif: ainsi, elle peut seulement apporter des informations sur une conception à un moment donné, sans que le "feed-back" apporté par la confrontation au dispositif puisse être pris en compte.

Au contraire, une épreuve visant à une étude clinique pourra être plus ouverte, le dépouillement pouvant prendre en compte de façon descriptive les différentes démarches. D'autre part, dans le cas d'une étude clinique, l'épreuve sur papier n'est qu'un point de départ: elle peut être complétée par un "*entretien*" qui consiste, à l'issue de l'épreuve , à demander aux élèves d'entrer en machine la solution qu'ils ont produite par écrit, puis de poursuivre leur recherche sous forme de travail dirigé. Ceci permet, à partir d'enregistrement au magnétophone des interventions des élèves, et des indications orales qui leur sont données, ainsi que de la trace écrite des différents essais de programme, erreurs et résultats d'exécution, de mieux prendre en compte la démarche de résolution des élèves, et ses évolutions éventuelles.

4.2 La méthodologie adoptée pour chacun des axes de l'étude expérimentale

Dans le cas de notre thèse, il est clair que le cadre d'analyse était à construire. Les premières observations ont donc été contemporaines de l'élaboration des premières hypothèses. Par la suite, le cadre d'analyse s'est affirmé, mais nous avons voulu continuer à prendre en compte les démarches de résolution des élèves, et amorcer une étude de la situation pédagogique. Ceci explique qu'une grande part de l'étude expérimentale de cette thèse soit de type clinique, à une exception près.

4.2.1 Premières observations cliniques sur les chaînes de caractères

Les premières observations (objectifs énoncés en 3.1.1) menées au cours de l'année scolaire 86-87 ont été de nature clinique. Nous n'avons pu, en effet, élaborer que très progressivement un cadre d'analyse pour l'interprétation des difficultés rencontrées par les élèves au cours des séances de travaux dirigés par le professeur et dans l'épreuve sur papier (EPR1), que nous leur avons fait fait passer au cours du mois de janvier (voir chapitre 6). Le petit nombre d'élèves dans la classe (11 à 13) nous a permis de classer finement ces difficultés, de les mettre en relation avec l'enseignement reçu et de repérer certains axes. Nous avons vraiment commencé à les situer après le dépouillement de séries d'*entretiens* que nous avons fait passer à trois groupes d'élèves.

Par la suite, dans la même classe, nous avons mené l'observation EPR2 (voir chapitre 7) de la même façon que EPR1. Le cadre d'analyse que nous avions commencé à élaborer s'est révélé productif, car nous avons pu cette fois interpréter les résultats obtenus dès le dépouillement de l'épreuve sur papier, la série d'entretiens que nous avons fait passer à un groupe d'élèves venant seulement en confirmation.

4.2.2 Une étude quantitative sur les chaînes de caractères

Nous avons voulu ensuite vérifier auprès d'un nombre plus important d'élèves le caractère de généralité des *difficultés* rencontrées. C'est pourquoi, en fin d'année 87-88, nous avons fait passer dans trois classes de deux lycées différents une épreuve (EPR3) reprenant des items analogues à EPR1 et EPR2. L'évaporation des effectifs dans ces classes en fin d'année nous a permis seulement de recueillir une cinquantaine de réponses, la population concernée offrant cependant une diversité suffisante en matière d'enseignement reçu, de conditions pédagogiques, et d'aptitudes scolaires pour que les résultats soient significatifs. Nous avons pu en effet confronter les difficultés rencontrées à l'enseignement reçu et aux aptitudes générales. (Voir Chapitre **8**).

4.2.3 Le travail sur les booléens

Nous avons mené le travail sur les booléens (Objectifs énoncés ci-dessus en 3.1.3) de la même manière que les premières observations sur les chaînes (épreuve sur papier suivie d'entretiens). Cette méthode clinique s'est en effet révélée bien adaptée à la construction d'un cadre d'analyse des conduites des élèves, à partir d'hypothèses cognitives. L'objectif n'est pas en effet, l'exhaustivité, mais la mise en évidence de ce que le cadre d'analyse que nous avons mis en place pour les chaînes peut se généraliser à d'autres types.

4.2.4 Choix pour l'ingénierie (travail sur les problèmes à partir des difficultés repérées, tentative de caractérisation d'une situation pédagogique)

Le travail d'ingénierie didactique dont les objectifs ont été énoncés en 1.3, et que nous présentons au chapitre **9** est conçu comme une progression d'exercices devant faire travailler les élèves sur un dizaine de séances, d'une durée de une heure à une heure et demi sous forme de Travail Dirigé. Portant sur le structure de chaîne, et ne visant donc pas l'ensemble des objectifs de la classe de Seconde, cette progression trouve sa place à un moment où les élèves ont commencé à aborder la structure itérative. Nous avons fait une pré-expérimentation au cours de l'année scolaire 87-88, puis une observation précise au cours de l'année scolaire 88-89; voulant comprendre en profondeur les conduites et l'évolution d'élèves face à ces énoncés, et analyser la situation pédagogique (résolution par petit groupe en présence de l'ordinateur avec intervention d'un enseignant), nous avons, ici aussi, fait le choix d'une étude clinique. Ce type d'étude implique que les séances de résolution fassent l'objet d'un enregistrement précis (magnétophone + trace des écrans), que les interventions pédagogiques soient maîtrisées, et que l'environnement de travail habituel ne soit pas perturbé. Dans notre cas, travaillant seul à l'observation et au dépouillement, il n'était pas envisageable de suivre plus de deux groupes de deux élèves. Nous avons donc demandé au professeur de pouvoir travailler avec deux groupes constitués d'élèves particulièrement en difficulté dans les apprentissages antérieurs. L'étude réalisée (chapitre **10**) est marquée par les difficultés importantes (bien qu'assez diverses) que rencontrent ces quatre élèves. Elle constitue un premier repérage de l'évolution des difficultés, et des caractéristiques de la situation pédagogique.