Utilisation du cadre théorique de la planification pour la conception d'algorithmes complexes par des élèves de lycée

Marie-Noëlle Guy

Mémoire de Master Recherche en Didactique des Mathématiques

Directeurs de Mémoire : Jean-Baptiste Lagrange et Fabrice Vandebrouck

Utilisation du cadre théorique de la planification pour la conception d'algorithmes complexes par des élèves de lycée.

Marie-Noëlle Guy
<marie.guy@etu.univ-pari-diderot.fr>

Paris, Juin 2013

Remerciements

À Jean-Baptiste Lagrange, qui m'a dirigée, conseillée et encouragée durant toute ma recherche, y compris depuis l'autre bout de la Méditerranée.

À Fabrice Vandebrouck, qui a accepté de me superviser pendant l'absence de Jean-Baptiste.

À Dominique Laval, dont la collaboration précieuse m'a permis de mener à bien cette recherche.

À Mme Natta, Proviseur du lycée Évariste Galois de Sartrouville, qui a accepté que je mène mon expérimentation dans son établissement.

À Inès, Siegfried, Quentin et tous les autres élèves de Terminale S, spécialité mathématique, du lycée Évariste Galois de Sartrouville, qui ont été malgré eux mes cobayes et qui ont joué le jeu.

À mon conjoint, Raphaël, pour son avis d'expert, ses remarques pertinentes, ses relectures et surtout son soutient inconditionnel.

Merci.

Résumé : Ce travail utilise les cadres théoriques de la planification en psychologie de la programmation et de la théorie des situations pour étudier la conception d'algorithmes par des élèves de lycée. Une attention particulière est portée dans les situations d'apprentissages à la part du travail de conception laissé à la charge de l'élève. L'étude est menée autour de l'algorithme de Kaprekar, choisi parce que son codage en langage algorithmique impose une démarche de planification, à la différence de l'exécution manuelle.
Une première partie du mémoire étudie des ressources existantes proposant des situations d'apprentissage en classe de mathématiques en lycée autour de l'algorithme de Kaprekar. Elle montre comment le travail à la charge des élèves est souvent réduit, par absence de prise en compte de la planification. Une seconde partie propose et analyse a priori une situation d'apprentissage puis exploite une expérimentation. Elle montre comment un découpage de la situation en phases alternant discussion collective animée par le professeur et travail en groupe permet de laisser aux élèves la responsabilité de l'élaboration d'un plan et de modules.

Table des matières

Introduction

L'introduction d'une partie algorithmique dans le programme de l'enseignement des mathématiques au lycée en 2009 pose de nouvelles questions didactiques et remet à l'ordre du jour les problématiques qui s'étaient posées dans les années 80 avec la mise en place du plan Informatique Pour Tous.

On sent ainsi une volonté renouvelée de la noosphère d'initier à l'algorithmique et à la programmation une grande partie des élèves en abordant dès le lycée des éléments d'algorithmique jusque là enseignés uniquement post-bac. Bien que le programme ne vise que l'apprentissage des premières instructions élémentaires, le but final reste la compréhension et la conception d'algorithmes.

Dans notre travail, nous allons adopter un point de vue didactique en étudiant la conception et la programmation d'algorithmes complexes, au sens où ils nécessitent plusieurs étapes différentes de traitement, par des élèves de lycée.

Pour cela, nous allons nous baser sur des recherches effectuées dans les années 80 en psychologie de la programmation. Nous allons ainsi introduire une méthode descendante basée sur l'intégration de schémas de programmation par les élèves, afin qu'ils puissent concevoir de manière autonome un algorithme complexe à l'aide d'un plan de programme. Nous étudierons l'effet de cette méthode et le comparerons à d'autres situations de programmation dirigées par d'autres enseignants.

Dans un premier temps, nous allons exposer les bases de la recherche que nous avons menée. Dans une seconde partie, nous étudierons des traces de pratiques d'enseignants sur un algorithme particulier : l'algorithme de Kaprekar. Enfin, nous exposerons les résultats de notre propre expérimentation.

Chapitre 1

Bases de recherche

1.1 Travaux de référence en didactique de l'algorithmique en France

Nous analysons dans cette partie quelques travaux menés en didactique de l'algorithmique. La littérature est relativement mince à ce sujet. En effet, hormis durant une dizaine d'années dans les années 80 où une option informatique a été mise en place dans les lycées français, l'algorithmique n'était pas un objet d'enseignement des mathématiques avant la réforme du lycée de 2009, et n'a pas non plus fait l'objet de travaux spécifiques dans d'autres disciplines.

Nous étudierons tout d'abord des articles publiés dans les années 80 avant de nous intéresser à des travaux effectués dans les années 2000.

1.1.1 Travaux en didactique et psychologie de l'informatique dans les années 80

Signification et fonctionnement du concept de variable informatique chez des élèves débutants, Renan Samurçay [25]

Cet article a été publié en 1985 dans la revue anglophone *Educational Studies in Mathematics*, destinée aux chercheurs en didactique. Il est écrit par Renan Samurçay, psychologue cognitive française, ayant fait de nombreuses recherches en didactique de l'informatique ainsi qu'en psychologie ergonomique. En 1985 existait en France une "option informatique" au lycée, et l'auteur s'est penchée sur les différentes conceptions de la notion de variables que se font les élèves.

Renan Samurçay se place dans le cadre de l'analyse épistémologique et cognitive des concepts enseignés, en s'appuyant notamment sur les travaux de Vergnaud (1983).

Cet article vise à étudier la conception et le traitement de la notion de variable chez des élèves de seconde ayant suivi un enseignement de 15h de programmation (en PASCAL) durant lequel les notions de variables, d'affectation, de boucle, de lecture et d'écriture ont été introduites. L'auteur a choisi une analyse synchronique rendant compte des compétences des élèves à un moment précis. Elle a ainsi analysé les réponses de 26 élèves à un questionnaire consistant en quatre algorithmes à trous. Renan Samurçay a opté pour ce type de questions car d'une part il permet d'étudier les tâches de lecture et d'analyse d'algorithmes non écrits par le sujet ; d'autre part, une tâche de construction

d'algorithme n'aurait pu être menée à bien par suffisamment d'élèves débutant pour pouvoir en tirer des résultats significatifs.

L'auteur commence par justifier le choix de la notion de variable comme objet d'étude par son caractère central en programmation. Elle justifie également son choix d'étudier les variables dans une structure de boucle plutôt que lors de simple déclaration ou d'alternative : dans une boucle, le concept de variable informatique apparaît différent de celui de variable mathématique. Elle identifie trois opérations sur les variables dans une boucle, renvoyant à trois aspects cognitifs : la mise à jour, le test d'arrêt et l'initialisation. L'auteur identifie également deux types de variables intervenant dans les problèmes de programmation : celles qui sont des données explicites du problème et celles qui sont rendues nécessaires par la solution informatique.

Après avoir présenté sa méthodologie, Renan Samurçay expose ses hypothèses de travail. Elle fait l'hypothèse que plus le traitement informatique des variables s'éloigne de l'exécution "à la main", plus les élèves rencontrent de difficultés, comme lors de la gestion d'un invariant de boucle par exemple. L'auteur présente ensuite le questionnaire distribué aux élèves en identifiant trois types de tâches.

Trois questions portent sur l'initialisation des variables, variables d'accumulation ou compteurs, étant des données explicites ou non. L'auteur fait l'hypothèse que les initialisations des compteurs seront les mieux traitées, notamment dans le cas où ils n'ont que le rôle de compteur.

La construction du test d'arrêt est évaluée dans deux questions. Dans la première, la variable intervenant dans le test d'arrêt est à la fois une donnée explicite et un compteur dégressif. Dans la seconde, il s'agit d'un compteur explicite. L'auteur s'attend à ce que la seconde question soit mieux traitée que la première.

Enfin, le problème de la mise à jour des variables est traité dans une question. Les élèves ont à leur disposition un programme modèle qui divise un nombre par deux un certain nombre de fois et affiche le résultat final, et un programme à compléter dont on veut qu'il multiplie par trois le carré d'un nombre un certain nombre de fois. Il ne s'agit cependant pas d'une simple tâche de reproduction puisque les deux énoncés appartiennent à des domaines opératoires différents.

L'auteur effectue ensuite une analyse détaillée des résultats, question par question en identifiant des types d'erreurs. Globalement, l'initialisation est la tâche qui pose le plus de difficultés aux élèves. Ce point soulève un problème plus général : la difficulté des élèves à *faire une hypothèse sur l'état initial d'une variable, connaissant la transformation et l'état final* [29].

Par ailleurs, comme attendu, les variables d'accumulation sont moins bien traitées par les élèves que les autres. Les variables avec lesquelles les élèves sont le plus à l'aise sont encore les compteurs, ce qui peut être dû au fait que ce type de variable a été plus institutionnalisé. Il en est de même pour la structure de boucle REPEAT ... UNTIL Résultat.

D'autre part, les élèves privilégient la lecture à l'affectation pour l'initialisation ; l'auteur explique ce phénomène par une meilleure compréhension de l'instruction de lecture de la part des élèves ainsi que par un besoin d'instructions de communication.

Finalement, cet article est un état des lieux : il met en évidence des problèmes de conception de la notion de variable mais ne propose pas de solutions. Il conduit donc à des questionnements sur des stratégies visant l'acquisition de

cette notion.

Aquisition of programming knowledge and skills, Renan Samurçay et Janine Rogalski [27]

Cet article a été publié en 1990 dans la revue anglophone *Psychology of Programming*. Il est cosigné par deux psychologues cognitivistes françaises, Janine Rogalski et Renan Samurçay.

La date de publication de cet article laisse suggérer que les auteurs ont pu appuyer leurs recherches sur les expérimentations faites lors de la mise en place d'une option informatique au lycée en France dans les années 80.

Cet article est divisé en deux parties. Dans un premier temps, les auteurs établissent un cadre d'étude de la représentation du savoir en programmation. Dans une seconde partie, elles mettent en évidence les difficultés cognitives rencontrées lors de l'apprentissage de la programmation.

Ces deux parties se placent dans le cadre général de la psychologie cognitive et de l'ergonomie. La première partie présente le cadre qu'elles ont mis en place suite à l'observation de novices en programmation (adultes et adolescents ayant par ailleurs reçu une éducation généraliste). La seconde partie est une compilation et une analyse des travaux menés par des chercheurs didacticiens et cognitivistes dans le domaine de la programmation.

Tout d'abord, les auteurs introduisent leur sujet en précisant qu'elles vont se concentrer sur l'acquisition de savoir en programmation, qu'elles définissent comme la capacité des étudiants à résoudre des problèmes de programmation. Elles précisent que le public étudié est constitué d'adultes et d'étudiants novices en programmation, et que leur thèse est que le processus d'enseignement ne peut pas reproduire le processus réel de construction du savoir par un scientifique ou un professionnel.

Une première partie est alors consacrée à la description du cadre d'analyse mis en place. Celui-ci est constitué de quatre espaces : les objets cognitifs, la résolution de problème, la connaissance des structures et l'expérience et la pratique. Les auteurs expliquent que pendant le processus d'acquisition des savoirs, ces quatre espaces évoluent et de nouvelles interactions se créent. Elles mettent également l'accent sur le fait qu'au début de ce processus, les étudiants se réfèrent nécessairement à des expériences du quotidien. Elles utilisent pour cela les Systèmes de Représentation et de Traitement (SRT) introduits par Hoc [9] comme modèle des structures mentales que le sujet met en œuvre à un moment donné d'une tâche donnée. Elles montrent ainsi qu'un premier SRT peut être construit par analogie avec d'autres domaines de savoirs comme les mathématiques ; mais que de telles références peuvent avoir des effets néfastes et que des connaissances anciennes en programmation peuvent même constituer un obstacle et nécessiter un processus de désapprentissage.

Les auteurs se penchent ensuite sur la tâche de programmation. Elles expliquent que les deux dimensions importantes pour passer d'un problème du monde réel à un programme implémentable sont la représentation et le traitement. Il y a ainsi deux manières de résoudre un problème par la programmation :

- soit on commence par le traitement pour résoudre le problème dans le monde réel puis on le traduit pour obtenir un programme,
- soit on commence par représenter les objets du problème avant de le traiter pour le résoudre.

La plupart des études portant sur l'acquisition de compétences en programmation portent sur la dimension du traitement. Les auteurs avancent plusieurs raisons à cela : des raisons historiques, l'importance de la planification dans la conception des programmes et le rôle de l'organisation des actions dans un premier modèle de programmation. La représentation et les tâches liées aux structures de données sont ainsi laissées de côté puisqu'elles n'apparaissent, dans les langages impératifs, que pour des problèmes complexes. Cependant, dans des langages relationnels comme Prolog ou dans les langages orientés objet, ces difficultés concernent aussi les débutants.

Elles définissent les notions de schéma : structures utilisées dans le traitement des informations pour atteindre des objectifs à petite échelle, et de plan : ensemble organisé de procédures dynamiques liées à des schémas statiques. Les recherches sur l'élaboration de programmes peuvent alors être classées selon le point de vue qu'elles adoptent : celui des schémas (et de programmes vus comme du texte) ou celui des plans (et de la programmation vue comme une activité). Le premier point de vue est plus efficace pour l'analyse des tâches comme la compréhension ou la correction d'un programme, alors que le second permet d'analyser les tâches relatives à la conception d'un programme.

Dans une seconde partie, les auteurs présentent plusieurs types de difficultés cognitives rencontrées lors de l'apprentissage de la programmation.

Elles soulignent tout d'abord deux obstacles majeurs à l'apprentissage de la programmation : la programmation ne peut pas être comparée à une activité de tous les jours et le fonctionnement de la machine qui n'est pas transparent pour les novices. Elles présentent ensuite plusieurs notions entraînant des difficultés cognitives.

Tout d'abord, les représentations conceptuelles que se font les novices du dispositif sur lequel ils programment peuvent mener à des difficultés. Le type de langage utilisé est en jeu. En effet, dans les langages impératifs, un Système de Représentation et de Traitement assez limité peut être construit à partir des actions à effectuer par la machine, tandis que les langages comme Prolog impliquent des objets plus complexes comme des relations ou des valeurs de vérités.

Ainsi, la complexité du dispositif informatique augmente en fonction de la distance du contrôle et de la variété d'entités virtuelles(mémoire virtuelle, variables...) qui simulent des objets sans identité physique. Les langages assembleur, Prolog ou orientés objets font ainsi partie des plus difficiles à manipuler pour des novices.

Les structures de contrôle, plus souvent étudiées dans une approche calculatoire que dans une approche fonctionnelle, posent aussi des problèmes aux débutants.

Pour ce qui est des structures de test, plus elles sont précises (IF... THEN BEGIN ... END ELSE BEGIN... END), mieux elles sont comprises et utilisées par les étudiants. Par ailleurs, bien que des connaissances en logique et en mathématiques facilitent l'utilisation des structures conditionnelles, elles ne suffisent pas puisqu'il faut également prendre en compte le rôle du dispositif informatique.

L'utilisation de structures itératives nécessite l'identification et la construction d'un invariant de boucle (mise à jour), d'une condition de continuation (test) et l'initialisation des variables. La conception "spontanée" des novices est celle où le test est placé en fin de boucle. Par ailleurs, ils ont également tendance

à vouloir utiliser des variables différentes à chaque passage dans la boucle.

La récursivité met en jeu deux concepts : l'auto-référence et l'imbrication. Les débutants ont tendance à utiliser un modèle itératif pour la récursivité, qui est compatible avec la récursivité terminale mais pas avec les autres formes de récursivité. Ces difficultés sont dues au manque de situations analogues dans la vie quotidienne, ainsi qu'au caractère dynamique de l'itération qui constitue un obstacle à l'acquisition de ce nouveau schéma de programme. Une solution est peut-être de traiter des problèmes récursifs avant d'enseigner les structures itératives.

Les représentations mentales des données peuvent aussi être sources d'erreurs. En effet, la conception que les novices ont des variables est souvent inadéquate. La difficulté de gestion des variables est d'autant plus forte lorsque ce sont des variables "internes" au programme, c'est-à-dire lorsqu'elles ne sont pas présentes lors d'une exécution à la main. Par ailleurs, les débutants ne font pas clairement la différence entre les variables locales et les variables globales, ce qui nécessiterait une meilleure compréhension du fonctionnement de la machine.

Enfin, les méthodes, c'est-à-dire les aides pour la recherche de stratégies et la gestion de la résolution de problèmes d'une classe donnée, étaient jusque là utilisées uniquement pour la formation des programmeurs professionnels. Des études menées dans des domaines autres que la programmation indiquent que ces méthodes pourraient être enseignées aux novices dans certaines conditions : qu'ils aient déjà acquis une bonne alphabétisation informatique et que les tâches proposées soient suffisamment complexes pour justifier l'utilisation de ces méthodes.

Les auteurs concluent que l'acquisition de connaissances en programmation est un processus complexe. En effet, il conduit les débutant à construire un nouveau Systèmes de Représentation et de Traitement, et de mauvaises conceptions des notions mises en jeu peuvent entraîner des difficultés de conceptualisation.

D'autres travaux autour de la programmation

Dans son article *De faire à faire faire* [24], Samurçay met l'accent sur la difficulté de planifier des actions pour un dispositif. En effet, elle étudie une situation dans laquelle des élèves de seconde, novices en programmation, doivent transmettre des instruction à un opérateur humain muni de deux machines :

- une machine à calculer classique A qui ne peut faire que des additions,
- une machine à calculer B qui ne peut faire que l'addition +1,

afin de calculer le produit de deux nombres quelconques. Ainsi, bien que les sujets sachent parfaitement exécuter eux-même l'opération demandée, et que la compréhension du dispositif d'exécution ne présente pas de difficultés cognitives majeures, ils rencontrent des difficultés au niveau de la programmation : la planification des actions du dispositif d'exécution.

Ces difficultés de planification des actions pour un dispositif sont particulièrement présentes dans la programmation. Cela peut s'expliquer par le fait que l'élève se place dans un "schème familier" souvent inadapté pour écrire des procédures.

Ainsi, le sujet met en place des SRT (Système de Représentation et de Traitement) constitués de connaissances déclaratives (schémas de résolution, règles d'écriture...) et d'éléments procéduraux. Il s'agit ainsi de "machines mentales" que le sujet fait fonctionner pour anticiper les réponses du dispositif informatique dans une situation donnée [9]. Cependant, dans une situation connue, le

sujet possède déjà un SRT naïf qu'il est nécessaire de modifier afin de passer à un SRT informatique pour comprendre les objets mis en jeu par l'utilisation du dispositif [14].

Pour Hoc [10], la formation du programmeur intègre également l'enseignement de méthodes. Les méthodes descendantes qui préconisent de générer le programme en exprimant sa structure des super-structures aux sous-structures, ne semblent pas adaptées aux débutants car elles vont à l'encontre de leurs stratégies, mais peuvent être bénéfiques si le sujet possède déjà un répertoire pertinent de plans de programmes.

1.1.2 Travaux menés dans les années 2000

Introduire des éléments d'algorithmique et de programmation dans l'enseignement secondaire ? Une ingénierie didactique, Nguyen Chi Thanh et Annie Bessot [17]

Cet article a été publié en 2010 dans la revue Petit x. Il est cosigné par deux chercheurs en didactique, Annie Bessot et Nguyen Chi Thành.

Annie Bessot fait partie de l'IREM de Grenoble et du LIG (Laboratoire d'Informatique de Grenoble). Elle est aujourd'hui retraitée. Elle a mené de nombreuses recherches en didactique des mathématiques ainsi qu'en didactique de l'informatique. Elle a ainsi dirigé plusieurs thèses, dont certaines en cotutelle avec des universités vietnamiennes, notamment celle de Nguyen Chi Thành, intitulée *Étude didactique de l'introduction d'éléments d'algorithmique et de programmation dans l'enseignement mathématique secondaire à l'aide de la calculatrice* (2005)[16]. Nguyen Chi Thành est aujourd'hui chercheur à la Faculté de l'Éducation, Université Nationale du Viêt-nam à Hanoï.

La date de publication de cet article (2010), suggère qu'il a été publié dans le contexte de la mise en place des programmes de 2009 introduisant en France l'enseignement de l'algorithmique au lycée. En effet, la majeure partie de cet article s'appuie sur le travail effectué par Nguyen Chi Thành dans sa thèse de 2005.

Cet article comporte 3 grandes parties. Dans un premier temps, les auteurs présentent une étude épistémologique de l'algorithmique et de la programmation. Ils présentent ici une genèse de la machine ordinateur. Ils donnent ensuite les résultats d'une étude institutionnelle comparative entre la France et le Viêt-nam concernant la présence de l'informatique dans l'enseignement mathématique secondaire. Dans cette partie, ils étudient l'écologie de l'informatique. Enfin, ils présentent l'ingénierie didactique qu'ils ont mise en place, en se référant à des recherches menées dans le cadre de la psychologie cognitive, ainsi qu'aux travaux effectués en théorie des situations didactiques.

Chacune de ces 3 parties présente donc un cadre théorique mais également une méthodologie différents. En effet, la première partie consiste en une étude historique de l'évolution des machines de calcul. Dans la seconde partie, l'étude institutionnelle se fait à travers l'étude des programmes et des manuels des deux pays. Enfin l'ingénierie a été expérimentée auprès de 94 élèves de seconde et de première, en France et au Viêt-nam. Il s'agit d'un travail mené en classe. Les auteurs ont mis en place puis analysé la séance ainsi que les productions des élèves.

Dans la première partie, les auteurs étudient la genèse de la machine ordinateur en distinguant 3 étapes, symbolisées par 3 types différents de machines.

Ils se concentrent tout d'abord sur l'étude de la machine arithmétique. Il s'agit de la première mécanisation des calculs. Dans cette machine, on inscrit les nombres dans les cases mémoire de la machine et on lit le résultat. Cette machine ne garde donc pas en mémoire le résultat et ne possède pas de mémoire effaçable.

Les auteurs étudient ensuite la machine analytique de Babbage. La différence majeure avec la machine précédente est que la machine analytique possède une mémoire effaçable, ce qui permet de faire émerger les concepts de variable informatique et de boucle. Ada Lovelace a ainsi écrit un programme qui permet de calculer les nombres de Bernouilli. Dans cette machine, les instructions sont sur des cartes externes et ne sont pas stockées dans la mémoire de la machine.

Enfin, les auteurs se concentrent sur la machine ordinateur de Von Neumann, dans laquelle le programme est stocké dans la mémoire effaçable.

À la fin de cette première partie, les auteurs définissent les concepts d'algorithmique et de programme et établissent les liens entre ces notions.

Dans une seconde partie, les auteurs étudient la présences d'éléments d'algorithmique et de programmation dans l'enseignement. Ils remarquent tout d'abord que ces notions sont présentes dans l'enseignement supérieur, et qu'elles y ont été introduites par les traités de Kuntzmann[13], Knuth[12] et Horowitz et Sahni[11]. Si les deux premiers choisissent un enseignement basé sur la présence d'une machine de référence, le dernier opte pour un enseignement de l'algorithmique passant par celui d'un langage de programmation représentatif d'une classe de langages existants. C'est cette dernière stratégie qui domine l'enseignement de l'algorithmique actuel.

Les auteurs présentent ensuite les résultats d'une analyse institutionnelle comparative de l'introduction d'éléments d'informatique au Lycée en France et au Viêt-nam. En France, l'algorithmique est introduit au sein de l'enseignement des mathématiques (dans les domaines de l'analyse, des statistiques ou de l'arithmétique), tandis qu'au Viêt-nam, il est aussi présent en informatique, qui est considérée comme une discipline autonome.

Cependant, l'examen de 3 manuels français et de 3 manuels vietnamiens montre que si des algorithmes et des programmes sont présents dans les deux pays, en revanche, l'algorithmique et la programmation n'y sont pas enseignées.

Ainsi, l'algorithmique vit mal dans les deux institutions : au Viêt-nam car elle est isolé dans l'enseignement et que les instruments de calcul ne sont pas pris en compte ; en France, car les logiciels comme le tableur se chargent des calculs sans nécessiter d'écrire l'algorithme.

À la fin de cette seconde partie, les auteurs remarquent que les nouveaux programmes de 2009, influencés par le travail de la commission Kahane (2001), introduisent des notions d'algorithmique.

Dans la dernière partie, les auteurs décrivent l'ingénierie didactique qu'ils ont conçue. Ils choisissent la première stratégie présentée plus haut (enseignement basé sur la présence d'une machine de référence) en se basant sur la calculatrice non programmable. Cette calculatrice possède deux types de mémoire : visible (touches A, B, C ou touche Ans) et invisible (touches parenthèses). Pour l'ingénierie, les auteurs ont conçu une calculatrice générique : l'Alpro, qui permet d'enregistrer l'historique des touches actionnées.

L'exercice à résoudre est un problème de tabulation : «Soit f une fonction polynomiale de degré n. Calculer les images par cette fonction de nombres x_0, $x_1, \ldots\ x_k \ldots$ espacés d'un pas p et appartenant à l'intervalle $[a, b]$ avec $x_0 = a$.»

Pour cela, les élèves sont successivement placés dans trois situations. Dans la

première situation, les élèves ne peuvent utiliser que la calculatrice Alpro. Dans la seconde, ils ont en plus à leur disposition un robot, Calculator I, qui sait appuyer sur les touche d'Alpro et imprimer le résultat. Dans la troisième situation, les élèves disposent d'un robot Calculator II qui comprend les instructions de répétition. Ces situations représentent les 3 machines décrites plus haut.

Dans les trois situations, le but est d'écrire les messages les plus courts possibles pour effectuer le calcul.

L'analyse des séances et des productions des élèves montre que lors de la situation 1, peu d'élèves (3%) pensent à utiliser les mémoires de la calculatrice. Dans la situation 2, les élèves utilisent la mémoire effaçable, mais donne à Calculator I des instructions qu'il ne peut pas comprendre. La situation 3 permet de mettre en place des boucles mais les élèves rencontrent des problèmes dans la gestion du compteur.

Ainsi, les auteurs ont mis en évidence la nécessité de l'effaçabilité de la mémoire pour l'introduction de la notion de variable informatique, afin de la distinguer de celle de variable mathématique. Par ailleurs, la calculatrice est utilisée comme une machine arithmétique alors qu'elle possède une mémoire effaçable. Enfin, un travail mathématique est nécessaire à la programmation, notamment dans le cas des boucles.

Cette ingénierie permet d'introduire les notions de variable et de boucle, mais n'étudie pas la stabilisation de ces notions. Par ailleurs, le travail fait ici dans le domaine de l'analyse mériterait d'être développé dans ceux des statistiques ou de l'arithmétique.

1.2 Premières hypothèses de recherche et problématique

L'analyse des travaux en didactique de l'algorithmique que nous avons menée soulève de nombreuses questions. La récente mise en valeur de l'algorithmique dans les programmes de lycée donne tout leur sens à ces questions.

Tout d'abord, on peut supposer que les difficultés d'appréhension des objets informatiques tels que les variables ou les boucles mises en évidence par Samurçay [25] et par Nguyen [17] persistent aujourd'hui. Cependant, si ces difficultés ont été clairement identifiées dès les années 80, hormis l'ingénierie didactique proposée par Nguyen dans sa thèse [16], peu de remédiations ont été proposées. Si la mise en œuvre de telles remédiations semble être un problème pertinent et toujours ouvert, nous allons cependant nous intéresser à une problématique sensiblement différente.

En effet, nous allons nous pencher sur la question de la planification d'algorithmes complexes, au sens de Samurçay et Rogalski [27]. Ces auteurs définissent la programmation comme la réunion de plusieurs tâches : la construction de programmes, leur compréhension, leur modification et leur correction.

> Aquiring and developing knowledge about programming is a highly complex process. It involves a variety of cognitive activities, and mental représentations related to program design, program understanding, modifying, debugging (and documenting).[27]

Afin d'utiliser le cadre théorique défini par Samrçay et Rogalski, nous allons émettre une première hypothèse de recherche :

Hypothèse 1. *Dans nos recherches, nous assimilerons* algorithmique *et* programmation.

Pour correspondre au plus près à cette hypothèse, ainsi que pour d'autres raisons didactiques que nous étudierons plus tard, nous allons utiliser le logiciel de programmation AlgoBox. En effet, le langage de programmation utilisé par AlgoBox est relativement proche de la langue naturelle, et le fait qu'il suffise de cliquer sur des boutons réduit les erreurs possibles lors de l'écriture des instructions.

De plus, un texte écrit en AlgoBox peut être vu soit comme un algorithme si on s'intéresse à sa logique et à ses performances, soit comme un programme si on s'intéresse à son implémentation.

Nous introduisons ainsi une deuxième hypothèse de recherche :

Hypothèse 2. *Dans nos recherches, nous supposons que les élèves ne rencontrent pas de difficultés liées à la compréhension du langage.*

En effet, les élèves avec lesquels nous allons travailler sont habitués à utiliser AlgoBox (voir 3.1). Bien que n'étant pas des experts, ils ont dépassé le stade d'alphabétisation et ont assimilé les contraintes du langage informatique d'AlgoBox. Comme nous l'avons précisé plus haut, nous aurions pu nous intéresser aux difficultés liées au langage, mais nous avons décidé de les ignorer pour nous concentrer sur les problèmes liés à la planification d'algorithmes.

Ainsi, l'objectif de nos travaux va être de permettre aux élèves de construire et de programmer des algorithmes complexes, c'est à dire comportant plusieurs étapes de traitement. Pour cela, nous allons les aider à identifier des schémas leur permettant de planifier ces algorithmes.

Cela nous amène à poser une première question problématique qui guidera notre travail :

Comment la planification peut aider à la conception d'algorithmes complexes ?

Pour y répondre, nous initierons les élèves à une méthode de programmation descendante. Dans un premier temps, nous les inciterons à planifier l'algorithme afin qu'ils identifient les étapes de traitement nécessaires à son exécution. Dans un second temps, ils programmeront ces étapes afin de les intégrer comme schémas de programmes. Enfin, ils pourront programmer l'algorithme complexe étudié à l'aide des schémas avec lesquels ils se seront familiarisés.

Nous nous interrogerons aussi sur l'efficacité d'une telle méthode :

En quoi l'identification de schémas permet-elle de laisser plus de tâches à la charge de l'élève ?

Nous étudierons donc des expérimentations menées par d'autres enseignants afin de déterminer si la planification des étapes permet de laisser plus d'autonomie aux élèves.

1.3 Choix de la situation

Afin d'expérimenter la mise en place de schémas et de plans, nous avons cherché un algorithme complexe (présentant plusieurs étapes de traitement), mais ne présentant pas de difficulté majeure de compréhension, afin de ne pas rajouter d'obstacles non pertinents aux élèves.

C'est pour cela que nous avons choisi de travailler à partir de l'algorithme de Kaprekar.

1.3.1 Présentation de l'algorithme

Histoire

L'algorithme de Kaprekar est un algorithme découvert en 1949 qui agit sur des entiers. Il fut étudié par le mathématicien indien D. R. Kaprekar (1905 - 1988) pour les nombres de quatre chiffres, mais peut être généralisé à tous les nombres.

Description de l'algorithme

L'algorithme de Kaprekar consiste à associer à un nombre quelconque n un autre nombre $K(n)$ généré de la façon suivante :

On considère un nombre n, écrit dans une base quelconque (généralement la base 10). On forme le nombre n_1 en arrangeant les chiffres du nombre n dans l'ordre croissant et le nombre n_2 en les arrangeant dans l'ordre décroissant. On pose $K(n) = n_2 - n_1$.

On itère ensuite le processus avec $K(n)$.

Exemples

En partant du nombre 5294 (en base 10), on obtient $K(5294) = 9542 - 2459 = 7083$. En répétant le processus, $K(7083) = 8730 - 378 = 8352$. Puis, $K(8352) = 6174$. On constate que $K(6174) = 6174$ et que l'algorithme conduit alors à un nombre fixe.

Si on commence avec 634, on obtient successivement 297, 693, 594, 495, 495, etc. On obtient là aussi un nombre qui ne varie plus.

Avec 52, la séquence est la suivante : 52, 27, 45, 09, 81, 63, 27, etc. La séquence se répète.

Partant de 63954, on obtient 63954, 61974, 82962, 75933, 63954, 61974, etc. La séquence se répète.

Phénomènes remarquables

Pour tout nombre initial, l'algorithme de Kaprekar produit au final l'une des possibilités suivantes :

- 0 (dans les cas dégénérés où les nombres s'écrivent avec un unique chiffre)
- Un nombre constant
- Un cycle de nombres

Pour la base 10, les premières possibilités sont les suivantes :

Résultat	Nombre de chiffres	Remarques
0		Pour les nombres s'écrivant uniquement avec le même chiffre (cas dégénéré)
9, 81, 63, 27, 45...	2	Cycle
495	3	Constante (dans tous les cas non dégénérés)
6174	4	Constante (dans tous les cas non dégénérés)
5355, 59994	5	Cycle (dans 3002 cas)
62964, 71973, 83952, 74943...		Cycle (dans 43219 cas)
61974, 82962, 75933, 63954...		Cycle (dans 43770 cas)
420876, 851742, 750843, 840852,	6	Cycle (dans 841996 cas)
860832, 862632, 642654...		
549945		Constante (dans 1815 cas)
631764		Constante (dans 56180 cas)
7509843, 9529641, 8719722, 8649432,	7	Cycle (dans tous les cas non dégénérés)
7519743, 8429652, 7619733, 8439552...		
63317664	8	Constante (dans 556234 cas)
97508421		Constante (dans 2041186 cas)
64308654, 83208762, 86526432...		Cycle (dans 43200472 cas)
43208766, 85317642, 75308643, 84308652,		Cycle (dans 44202099 cas)
86308632, 86326632, 64326654...		

1.3.2 Exploitation en classe

Choix du niveau

Cet algorithme peut être exploité à différents niveaux, du collège au supérieur. En effet, comme on l'a vu dans sa présentation, cet algorithme engendre des phénomènes remarquables. C'est grâce à cela qu'il a été rendu célèbre par Martin Gardner dans le cadre des mathématiques récréatives. Par ailleurs, l'étude de cet algorithme a aussi été effectuée dans le cadre de l'association Maths.en.Jeans [15]. Les questions que suscitent ces phénomènes font de cet algorithme un support pour des activités d'apprentissage en classe.

Dans les années 1980, Gérard Chauvat s'appuie sur cet algorithme pour développer des activités en « Arithmétique élémentaire en classe de 5ème » utilisant la notion de classe d'équivalence.

Aujourd'hui, le travail de numération sous-jacent à la mise en œuvre de cet algorithme permet de l'exploiter dès le collège [5].

Par ailleurs, l'introduction de l'algorithmique dans les nouveaux programmes de lycée rend l'étude de cet algorithme particulièrement pertinente dans les classes de seconde, première et terminale. Il s'inscrit également dans les programmes des spécialités mathématiques ou informatique et sciences du numérique en terminale S.

Enfin, il est également intéressant de le mettre en œuvre dans les classes de l'enseignement supérieur, en cours d'arithmétique ou d'informatique. Jean-Claude Rauscher l'a également utilisé dans le cadre de la formation initiale des futurs professeurs. De plus, la deuxième épreuve d'admissibilité du concours de recrutement des professeurs des écoles 2010 comportait un exercice étudiant l'algorithme de Kaprekar [2].

Dans la suite de notre étude, nous choisirons de nous concentrer sur l'analyse de l'utilisation de cet algorithme à la fin du lycée (terminale S). En effet, à

ce niveau-là, les élèves ont suffisamment de connaissances arithmétiques pour comprendre l'algorithme et amorcer un ébauche de preuve.

D'autre part, l'étude des algorithmes s'inscrit dans les nouveaux programmes du lycée :

> Dans le cadre de cette activité algorithmique, les élèves sont entraînés à :
> - décrire certains algorithmes en langage naturel ou dans un langage symbolique ;
> - en réaliser quelques-uns à l'aide d'un tableur ou d'un programme sur calculatrice ou avec un logiciel adapté ;
> - interpréter des algorithmes plus complexes.
>
> (...)
>
> L'algorithmique a une place naturelle dans tous les champs des mathématiques et les problèmes posés doivent être en relation avec les autres parties du programme (analyse, géométrie, statistiques et probabilités, logique), mais aussi avec les autres disciplines ou le traitement de problèmes concrets. [3]

Les élèves ont également, dans le cadre du programme, les moyens de mettre en œuvre l'algorithme de Kaprekar :

> À l'occasion de l'écriture d'algorithmes et programmes, il convient de donner aux élèves de bonnes habitudes de rigueur et de les entraîner aux pratiques systématiques de vérification et de contrôle.
>
> *Instructions élémentaires (affectation, calcul, entrée, sortie)*
>
> Les élèves, dans le cadre d'une résolution de problèmes, doivent être capables :
> - d'écrire une formule permettant un calcul ;
> - d'écrire un programme calculant et donnant la valeur d'une fonction ;
> - ainsi que les instructions d'entrées et sorties nécessaires au traitement.
>
> *Boucle et itérateur, instruction conditionnelle*
>
> Les élèves, dans le cadre d'une résolution de problèmes, doivent être capables de :
> - programmer un calcul itératif, le nombre d'itérations étant donné ;
> - programmer une instruction conditionnelle, un calcul itératif, avec une fin de boucle conditionnelle. [3]

Énoncé

Nous présenterons aux élèves le programme de calcul suivant (le choix de cet énoncé sera justifié plus tard, cf 1.4.2) :

1. Choisir un nombre entier de trois chiffres.
2. Former le nombre obtenu en arrangeant les chiffres du nombre choisi en 1. dans l'ordre croissant.
3. Former le nombre obtenu en arrangeant les chiffres du nombre choisi en 1. dans l'ordre décroissant.
4. Calculer la différence des nombres obtenus en 2. et 3.
5. Recommencer (à partir de l'instruction 2.) avec le résultat obtenu en 4, jusqu'à obtenir un nombre déjà obtenu.

La tâche sera alors de construire un algorithme effectuant ce programme de calcul et de le programmer sur AlgoBox (voir le Document élève A). Ce faisant, les élèves seront amenés à effectuer des conjectures sur le comportement de ce programme de calcul, qui pourront être prouvées à l'aide de l'ordinateur et algébriquement par la suite (voir travaux de Dominique Laval).

1.4 Analyse a priori

1.4.1 Objets d'apprentissage mis en jeu dans cette situation didactique

L'établissement d'une conjecture nécessite la compréhension de l'algorithme de Kaprekar. Pour cela, l'élève doit savoir *effectuer un programme de calcul* et *maîtriser la numération de position* afin de former les nombres des étapes 2. et 3. Ces compétences, normalement acquises au niveau où nous avons choisi de placer notre étude, peuvent être un objectif visé si l'on veut exploiter cet algorithme au collège.

La construction de l'algorithme demande d'approfondir les savoirs visés par l'établissement de la conjecture. En effet, la programmation de l'algorithme nécessite de comprendre chaque procédé utilisé, notamment le *passage du nombre à la suite de ses chiffres* par l'utilisation de la division euclidienne par exemple (pour plus de détails, voir 1.4.3). De plus, l'utilisation d'un *algorithme de tri* s'avère nécessaire. Ici se posent également des problèmes liés au passage d'une procédure manuelle à un programme effectué par une machine. En effet, la question du tri est transparente en procédure manuelle (en particulier sur des nombres à seulement trois chiffres), alors qu'elle nécessite d'être clairement explicitée pour pouvoir être programmée sur une machine.

Bien que la preuve de la conjecture (Si le nombre choisi comporte au moins deux chiffres distincts alors le résultat obtenu est 495, sinon, on obtient 0) ne soit pas traitée dans le cadre de nos travaux, mais exploitée dans ceux de Dominique Laval, nous pouvons remarquer qu'elle présente deux objectifs distincts.

En effet, les élèves peuvent choisir de démontrer la conjecture en examinant tous les cas à l'aide de l'outil informatique (les 1000 cas sont traités très rapidement par une machine), ce qui introduit la notion de preuve non explicative.

Ils peuvent également résoudre le problème de manière algébrique :

Soit N le nombre choisi en 1. tel que $N = \overline{abc}$ en base 10.

On peut supposer $a \geq b \geq c$.

Soit G le nombre calculé en 2. et P le nombre calculé en 3. On a $G = \overline{abc}$ et $P = \overline{cba}$.

Soit D le nombre calculé en 4. :

$$\begin{aligned} D &= G - P \\ &= \overline{abc} - \overline{cba} \\ &= 100 \times a + 10 \times b + c - (100 \times c + 10 \times b + a) \\ &= 99a - 99c \\ &= 99(a - c) \end{aligned}$$

Donc le nombre D est un multiple de 99.

On sait que $(a - c)$ est un nombre entier compris entre 0 et 9 car $a \geq c$. Il ne reste donc que 10 cas à traiter.

Cette solution permet d'introduire *le calcul algébrique comme méthode de résolution de problèmes arithmétiques* ainsi que le *raisonnement par disjonction de cas.*

Sans aller jusque là, un travail simple sur le mécanisme de la soustraction permet d'établir qu'après une itération de l'algorithme, le chiffre des dizaines est nécessairement 9.

1.4.2 Choix des variables didactiques

Nous avons choisi de faire travailler les élèves avec des nombres à trois chiffres. Ce choix est motivé par plusieurs raisons. Tout d'abord, nous évitons de traiter le cas des nombres à deux chiffres, d'une part parce que le nombre d'entiers à tester est faible et que le travail peut être effectué à la main par les élèves ; d'autre part car il nous semble plus intéressant de mettre en évidence un point stationnaire pluôt qu'un cycle. En effet, les élèves sont plus familiers des points stationnaires ; lors de l'étude de la suite de Syracuse par exemple, le fait que la suite converge vers un cycle perturbe certains élèves qui ont du mal à admettre qu'une fois que l'on est entré dans le cycle, il se répète indéfiniment.

Ainsi les nombres à trois ou quatre chiffres nous semblent les plus pertinents. Pour les nombres à 5 ou 6 chiffres, on obtient plusieurs comportements différents (points stationnaires et cycles), ce qui nous semble être une difficulté supplémentaire et superflue pour les élèves.

Afin de pouvoir effectuer une preuve algébrique au niveau lycée, nous préférons choisir les nombres à trois chiffres, qui présentent une distinction de cas avec seulement 10 cas.

1.4.3 Analyses des différentes stratégies

Lorsqu'il est confronté à l'énoncé de ce problème, l'élève peut mettre en place plusieurs stratégies. Tout d'abord, différents logiciels de calculs peuvent s'offrir à lui : tableur, calculatrice, AlgoBox... Ainsi, selon ses habitudes mais aussi selon le contrat didactique mis en place, il peut s'orienter vers l'une ou l'autre de ces options. Parfois, l'enseignant peut en favoriser une dans l'énoncé : le tableur, choisi par le manuel Math'x et par une enseignante de Franche-Comté [5] ou AlgoBox, choisi par plusieurs autres enseignants [5]. En fonction du logiciel choisi, les contraintes de programmation ne sont pas les mêmes.

En effet, comme nous l'avons vu plus haut, pour la programmation de cet algorithme, l'élève est confronté à deux problèmes distincts : le passage du nombre aux chiffres (et inversement), et le problème du tri.

Étudions tout d'abord différentes stratégies pour le passage du nombre à ses chiffres.

Une première tentative est d'effectuer des soustractions successives pour obtenir les chiffres de $\overline{abc}$. Ainsi, pour extraire le chiffre des unités, on obtient un algorithme du type :

```
Variables
        x est du type nombre
Debut
        Saisir x
        Tant que x>10 faire
                x ← x-10
        FinTantQue
        Afficher x
```

`Fin`

À partir d'un tel algorithme, un travail sur la numération peut être mené afin d'aboutir à deux raffinements possibles :

- Une première stratégie est de voir a dans $\overline{abc}$ comme la partie entière de la fraction $\dfrac{\overline{abc}}{100}$. On y a accès grâce à la fonction `floor` dans AlgoBox et ENT dans un tableur. Il faut ensuite recommencer l'opération avec $\dfrac{\overline{abc}-100a}{100}$ pour obtenir b, puis avec $\dfrac{\overline{abc}-(100a+10b)}{100}$ pour obtenir c.
- On peut également prendre le problème dans l'autre sens, en considérant c comme le reste de la division euclidienne de $\overline{abc}$ par 10. Cela est possible grâce à la fonction `%` d'AlgoBox et MOD du tableur. On calcule ensuite b comme le reste de la division euclidienne de $\overline{abc}-c$ par 100, et c comme $\dfrac{\overline{abc}-c-10b}{100}$.

Une autre stratégie ne fait pas intervenir les connaissances en numération mises en évidence dans la partie 1.4.1. Il s'agit en effet de voir le nombre comme une chaîne de caractères dont on veut extraire des sous-chaînes. Il faut pour cela pouvoir passer du type nombre au type chaîne et vice versa facilement. Cela est possible avec le tableur. La fonction STXT permet d'extraire une chaîne de caractères et la fonction CNUM transforme du format texte au format numérique. Sur AlgoBox en revanche, bien que la fonction `.substr` permette d'extraire une sous-chaîne et `.toString` convertisse un nombre en chaîne, il n'y a pas de façon aisée de passer d'un caractère numérique d'une chaîne au nombre associé (il est nécessaire de passer par son code ASCII avec l'instruction `.charCodeAt`). Cette stratégie parait donc peu pertinente pour un traitement avec AlgoBox.

De nombreuses stratégies s'offrent également à l'élève pour le tri des chiffres.

Une première stratégie, n'utilisant que des outils de base, est de tester les 6 cas (a<b<c, a<c<b, b<a<c, b<c<a, c<a<b et c<b<a) avec 6 instructions `Si...Alors...`, et d'affecter les variables P et G en conséquence. On peut également programmer des variantes en utilisant des instructions du type `Si...Alors...Sinon`, ce qui réduit le nombre de tests.

On peut également construire d'autres méthodes de tri en stockant les chiffres du nombre de départ dans des listes (comme suggéré sans le document pour l'enseignant produit par l'IREM de Franche-Comté [5]). Tout d'abord, le fait de traiter des listes à trois éléments nous permet de contourner le problème en utilisant les fonctions `min` et `max` (se trouvant dans AlgoBox et dans le tableur), et en calculant le "terme du milieu" comme la somme des termes de la liste à laquelle on a retranché le min et le max. On peut également utiliser d'autres méthodes de tri sur les listes (tris à bulles, par sélection ou par insertion par exemple), mais qui nécessitent de bien maîtriser l'échange de deux variables à l'aide d'une variable auxiliaire, et qui nous semblent hors de portée au niveau étudié.

Cependant, dans le cas où l'élève utilise un tableur, le problème du tri peut être évité grâce à la fonction GRANDE.VALEUR qui renvoie la $k$ième plus grande valeur d'une série de nombres. En outre, dans un tableur, seules les affectations de variables sont à gérer, la boucle étant remplacée par le fait "d'é-

tendre la case" (recopie vers le bas avec des références relatives).

Pour la dernière étape, la reconstruction du nombre à partir de ses chiffres, la stratégie consiste à former à partir des chiffres a, b, et c le nombre $\overline{abc} = 100 \times a + 10 \times b + c$.

Enfin, il reste à préciser une dernière stratégie pour la conception de l'algorithme, totalement différentes de celles étudiées précédemment : plutôt que de travailler avec des nombres, on peut choisir de travailler uniquement avec des chiffres.

En effet, cela permet d'éviter le problème du passage du nombre aux chiffres, mais en soulève d'autres. Tout d'abord, cet algorithme comportant une boucle, il faut bien faire attention à avoir le même type de variables en entrée et en sortie. Ainsi, si on choisit d'entrer trois chiffres, il faut renvoyer trois chiffres et non pas un nombre à trois chiffres. Pour éviter complètement le passage du nombre aux chiffres, il faut donc effectuer tout le calcul sur les chiffres sans jamais reconstituer le nombre. Cela nécessite de construire la soustraction en travaillant sur les chiffres, et donc de gérer le système de retenues.

Afin d'exploiter les connaissances des élèves en termes de numération et de leur permettre d'identifier clairement plusieurs étapes distinctes, nous choisissons de ne pas favoriser ce type de stratégie. Grâce à l'énoncé précisant *Choisir un nombre entier de trois chiffres*, les élèves seront guidés vers les stratégies précédentes travaillant sur des nombres à trois chiffres.

1.4.4 Analyses théoriques de la situation

Dans cette partie, nous analysons le potentiel didactique de la situation que nous avons choisi de mettre en place. Après avoir identifié les registres de représentation ainsi que les cadres mis en jeu dans cette situation, nous étudierons la dialectique outil-objet de l'algorithmique qu'elle met en valeur.

Enfin, nous verrons comment cette situation est adaptée à l'étude de la planification d'algorithmes complexes avant de voir pourquoi la théorie des situations nous semble être un bon outil pour l'étudier.

Registres de représentation sémiotique du nombre mis en jeu dans la situation

Cette situation didactique est basée sur des changements de *représentation sémiotique* [8] des nombres. En effet, dans l'étape 1, le nombre $\overline{abc}$ est considéré comme un élément de l'ensemble $\{0, 1, \dots 999\}$, mais pour construire les nombres des étapes 2. et 3, il est nécessaire de se représenter le nombre comme un triplet de chiffres de l'ensemble $\{0, 1, \dots 9\}^3$, avant de repasser dans l'ensemble $\{0, 1, \dots 999\}$.

Par ailleurs, l'utilisation, dans la preuve comme dans l'algorithme, de lettres pour représenter les chiffres du nombre choisi, donne une représentation du nombre dans le registre algébrique. C'est la combinaison du registre numérique et du registre algébrique qui permet l'introduction du calcul algébrique dans notre situation.

Analyse des cadres didactiques de la situation

La situation fait naturellement intervenir plusieurs cadres didactiques. En effet, le problème semble a priori se situer dans un cadre arithmétique faisant intervenir l'écriture de nombres dans le système de numération en base 10, mais

le fait de se retrouver face à un procédé de calcul donne envie d'observer cette situation dans un cadre plus algorithmique. Ce jeu de cadre se fera d'autant plus facilement que les élèves sont, grâce aux nouveaux programmes, incités à *mettre au point des solutions algorithmiques* (Ressources pour la classe de seconde – Algorithmique, Juin 2009). Ce changement de point de vue amène les élèves à se poser de nouvelles questions, notamment au sujet du système décimal :

- Comment isoler le chiffre des unités d'un nombre à 3 chiffres ? Celui des dizaines ? Des centaines ?
- Inversement, comment écrire un nombre à partir de ses chiffres ?

Ce changement de cadre permet donc une consolidation des connaissances sur la numération en même temps qu'il pose de nouvelles questions, intrinsèques au traitement algorithmique du problème cette fois :

- Comment ranger trois chiffres a, b et c dans l'ordre croissant ?

Par ailleurs, le problème de la démonstration de la conjecture fait basculer la situation dans le cadre algébrique. En effet, le besoin de généralisation conduit les élèves à utiliser des lettres pour représenter les chiffres et effectuer le procédé de calcul.

Dialectique Outil-Objet de l'algorithmique

Dans notre situation apparaissent naturellement les aspects outil et objet de la notion d'algorithme. En effet, l'objet de notre problème est un algorithme, il s'agit bien d'un *objet culturel ayant sa place dans le savoir mathématique* [7] ou encore dans la science informatique. Ce même algorithme intervient, notamment une fois programmé grâce à l'outil technologique qu'est l'ordinateur, comme un outil pour la démonstration de l'exercice 2. En effet, dans la première preuve, non explicative, envisagée, c'est uniquement grâce à l'exécution de l'algorithme sur ordinateur qu'on prouve la conjecture envisagée. Dans la seconde démonstration, la preuve des 10 cas distincts peut également être faite en exécutant cet algorithme pour ces 10 cas.

Lorsqu'ils abordent ce problème, les élèves de lycée peuvent compter sur un *ancien* [7] conséquent. En effet, ils savent déjà, depuis le primaire, appliquer à la main des programmes de calcul. Ils peuvent ainsi tester plusieurs fois l'algorithme et établir la conjecture. Par ailleurs, les compétences en numération nécessaires à la programmation de l'algorithme (écriture en base 10) sont au programme du cours de spécialité mathématiques de la classe de terminale S.

La question de la programmation de l'algorithme afin de le tester un grand nombre de fois soulève de nouvelles questions liées au langage. En effet, pour écrire et programmer l'algorithme, il est nécessaire de pouvoir généraliser les étapes de calcul, faites auparavant à la main sur un exemple précis, à un nombre à trois chiffres quelconque. L'introduction de lettres et donc le recours à l'écriture algébrique est alors nécessaire. Les élèves ont donc recours à un *nouveau implicite* [7] pour traiter le problème.

Ces nouvelles connaissances sont alors prêtes à être institutionnalisées, par exemple pendant la démonstration en explicitant l'égalité : $\overline{abc} = 100 \times a + 10 \times b + c$.

Schémas et plans

Comme nous l'avons vu lors de l'étude des stratégies possibles (cf 1.4.3), la conception de l'algorithme de Kaprekar nécessite l'identification d'étapes de calcul propres au fonctionnement du dispositif informatique (machine + langage).

Ces étapes de calcul (décomposition du nombre en chiffres, tri des chiffres et reconstitution du nombre) peuvent être vues comme des schémas (au sens de Samurçay et Rogalski : *standard structures that can be used to achieve small-scale goals* [22]). En favorisant l'identification et la conception de ces schémas, la planification de l'algorithme devrait être plus accessible pour les élèves.

Ainsi, nous voulons favoriser l'utilisation d'une méthode descendante en suggérant aux élèves de commencer par étudier la structure générale de l'algorithme avant de s'intéresser à la programmation des sous-structures nécessaires. En programmant ces étapes intermédiaires de calcul, les élèves pourront ainsi s'approprier des schémas de programmes qui pourront être réinvestis au moment de la conception de l'algorithme complexe.

Utilisation de la théorie des situations didactiques

Il semble que cette situation didactique présente un certain potentiel d'adidacticité. En effet, on pourrait présenter le problème en ne donnant que l'énoncé de l'algorithme de Kaprekar et en laissant les élèves réagir. La dévolution serait alors délicate, peut-être qu'il serait préférable de faire travailler les élèves en petit groupes afin que leur curiosité soit éveillée par le fait que tout le monde obtienne le même résultat après quelques itérations.

Cependant, le contrat didactique présent dans les classes de lycée habituées à travailler l'algorithmique permet aux élèves de penser à programmer l'algorithme afin de le tester sur un ordinateur. Ce programme jouerait alors le rôle de *milieu* [4] en interagissant avec l'élève. Ceci conduirait les élèves vers une situation de formulation de la conjecture, voire même vers une preuve s'ils testent tous les entiers à trois chiffres.

Il est possible que cette preuve « par ordinateur » ne satisfasse pas tous les élèves et les pousse vers une situation de validation durant laquelle ils émettraient des jugements sur les formulations précédentes. Le recours au calcul algébrique pourrait alors être une solution à l'établissement d'une preuve « plus explicative ». Ce travail sur la preuve sera effectué par Dominique Laval dans le cadre de son travail de thèse.

Notre but étant ici de laisser le plus d'initiatives à la charge de l'élève possible, il nous semble que la théorie des situations développées par Brousseau permet d'observer le travail de l'élève confronté à un problème nouveau. Ceci nous conduit à exprimer une nouvelle hypothèse de recherche.

Hypothèse 3. *La Théorie des Situations est un outil adapté à l'analyse de situations de conception d'algorithmes complexes.*

1.5 Méthodologie

Dans un premier temps, nous avons tenté de nous faire une idée de l'état de l'art dans le domaine de la didactique de l'algorithmique (cf 1.1). Cette étude nous a permis de nous placer dans la théorie de la planification développée par Rogalski et Samurçay [22] pour nos recherches. Nous avons ensuite choisi une situation adaptée à l'observation de la planification avant d'étudier les enjeux didactiques de cette situation.

Dans un second temps, nous allons mener une étude de manuels et d'énoncés relatifs à l'écriture et la programmation de l'algorithme de Kaprekar. Pour cela, nous nous appuierons sur les énoncés donnés par les enseignants à leurs élèves, mais aussi sur les traces de pratiques que ces enseignants ont publiées, notamment dans le cadre de travaux IREM. Dans cette étude, nous nous pencherons tout particulièrement sur les tâches laissées à la charge de l'élève lors de la conception de l'algorithme, et sur celles à la charge de l'enseignant.

Enfin, nous mènerons notre propre expérimentation. Pour cela, nous concevrons un énoncé permettant d'aider les élèves à concevoir des schémas de programmes qu'ils utiliseront pour l'écriture de l'algorithme de Kaprekar.

Cette situation sera expérimentée dans un groupe de terminale S spécialité mathématiques au lycée Évariste Galois de Sartrouville (92). Bien que le lycée accueille un public assez hétérogène, ce groupe est plutôt d'un bon niveau (6 élèves sont présentés au Concours Général).

L'expérimentation consistera en deux séances (une de deux heures et une de 80 minutes). Ces deux séances seront dirigées par Marie-Noëlle Guy et Dominique Laval en sera observateur. Des enregistrements sonores (de la classe et de trois groupes d'élèves) seront effectués. Les traces relevées seront donc ces enregistrements ainsi que les travaux papier des élèves et leurs programmes AlgoBox.

Chapitre 2

Étude de manuels, d'énoncés et de traces de pratiques

Le problème de l'écriture de l'algorithme de Kaprekar n'est traité que dans peu de manuels, probablement car considéré trop complexe pour être présenté à des élèves de lycée puisqu'il comporte plusieurs étapes.

Cependant, plusieurs enseignants ont proposé ce type de situation à leurs élèves. Certains ont ainsi publié sur des sites d'IREM des traces de leurs pratiques.

2.1 Étude du problème proposé par le manuel Math'x

Nous étudierons les parties A et B concernant la programmation de l'algorithme de Kaprekar et laisserons de côté la partie C traitant de la démonstration.

Tout d'abord, l'énoncé de la *recette de Kaprekar* est très détaillé. En effet, on précise qu'il faut tout d'abord *extraire les chiffres* du nombre choisi. Par ailleurs, après une première partie permettant aux élèves de prendre un premier contact avec l'algorithme et de l'exécuter à la main, les élèves sont invités à utiliser un tableur. On ne laisse pas le choix du logiciel aux élèves, et la manuel annonce que l'objectif de la partie B n'est non pas de faire exécuter l'algorithme, mais de *créer une feuille de calcul comme celle dont la copie d'écran figure ci-après.*

L'étape d'extraction des chiffres est alors extrêmement détaillée. En effet, on ne demande aux élèves que de *justifier que le chiffre des unités d'un nombre est le reste de la division par 10 de ce nombre*, et l'utilisation de la fonction MOD est suggérée dans l'encadré *Aide tableur*.

L'étape de tri est également très guidée puisqu'on suggère aux élèves d'utiliser la fonction GRANDE.VALEUR.

Il ne reste à la charge de l'élève que la reconstitution du nombre.

Cette activité est donc extrêmement guidée par l'énoncé.

2.2 Travaux menés par l'IREM de Franche-Comté

Le groupe Lycée de l'IREM de Franche-Comté a mené en 2010 un travail d'équipe autour de l'algorithme de Kaprekar sous la responsabilité d'Alain Parmentelat. Nous allons étudier les énoncés et traces de pratiques publiés par les

enseignants sur le site de l'IREM de Franche-Comté [5].

2.2.1 Travaux de Françoise (voir Annexe D.2)

Cette enseignante a proposé aux élèves une activité en quatre temps, un premier exercice à préparer en temps libre suivi d'un premier bilan en classe, puis un second exercice donné en temps libre et enfin un dernier bilan en classe.

L'énoncé que l'enseignante propose aux élèves ne porte tout d'abord pas sur l'écriture ou la programmation de l'algorithme de Kaprekar. En effet, il s'agit plus d'une première prise de contact avec l'algorithme pendant laquelle les élèves sont invités à "faire tourner l'algorithme" (à la main), ainsi qu'à émettre une conjecture. Cependant, la première question

(a) Quelles sont les variables informatiques utilisées dans cet algorithme?

suggère que les élèves s'engagent déjà sur la voie de la conception de l'algorithme. Il semble pourtant peu probable que les élèves pensent à ce stade à l'introduction de variables pour stocker les chiffres du nombre.

Lors du premier bilan en classe, l'enseignante propose aux élèves de démontrer la conjecture émise. À ce moment, un groupe d'élève propose de tester tous les cas possibles avec un tableur ; et ce n'est qu'à ce moment-là que la question de la programmation de l'algorithme est abordée.

Les élèves sont alors très guidés par l'enseignante qui leur présente plusieurs fonctions du tableur. Elle les laisse alors travailler individuellement avec le tableur. Les élèves ont alors réussi à extraire les chiffres du nombre et à les trier à l'aide des fonctions suggérées par l'enseignante. Cependant, le passage des chiffres au nombre (qui n'avait pas été développé par l'enseignante) a posé de nombreuses difficultés aux élèves.

Les élèves ont alors eu à terminer l'algorithme chez eux.

Ainsi, l'enseignante a pris à sa charge les deux premières étapes : passage du nombre aux chiffres et tri des chiffres, en suggérant l'utilisation de certaines fonctions du tableur. Par ailleurs, les traces de l'enseignante n'indiquent pas si c'est elle ou les élèves qui ont eu l'idée de la décomposition en chiffres.

2.2.2 Travaux en première S (voir Annexe D.3)

Cet enseignant a choisi de travailler autour de l'algorithme de Kaprekar pendant 3 séances durant lesquelles les élèves travaillaient en demi-classe par groupe de 2 ou 3.

L'objectif de l'enseignant est que les élèves programment l'algorithme de Kaprekar en AlgoBox.

La première séance est constituée d'une première rencontre avec l'algorithme, et de la programmation d'un algorithme de tri de trois chiffres. Malgré les difficultés liées à l'alphabétisation rencontrées (confusion entre les instructions Lire et Écrire), un unique groupe d'élève parvient à écrire un algorithme distinguant les 6 cas à l'aide de structures Si... Alors ... FinSi. Cependant, l'enseignant suggère d'utiliser des instructions Sinon, et laisse les élèves terminer l'algorithme à la maison. Mais les traces ne présentent pas les travaux effectués par les élèves, et l'enseignant précise qu'il distribue un corrigé à la séance suivante. Ainsi, bien que les élèves aient cherché à construire cet algorithme, l'algorithme final est fourni par l'enseignant.

La seconde séance porte sur la décomposition du nombre en chiffres. Cela semble poser de nombreuses difficultés aux élèves. Selon l'enseignant, deux idées *émergent finalement* : tant que $x > 10$, x prend la valeur $x - 10$ et l'utilisation

de la partie entière pour trouver le chiffre des centaines. Ici, on ne connaît pas réellement les interventions de l'enseignant, et on ne peut donc pas vraiment savoir à quel point ces idées viennent des élèves. Par ailleurs, c'est l'enseignant qui propose une troisième stratégie utilisant la division euclidienne.

La dernière séance à pour but de reconstituer l'algorithme à partir des travaux faits précédemment. Contrairement à ce qui a été observé dans la classe de Françoise, cette fois-ci la reconstitution du nombre ne pose que peu de problèmes. Nous supposons que cela est du au fait que le passage du nombre aux chiffres a ici été traité en profondeur et d'un point de vue de la numération, alors que Françoise avait utilisé un artifice :

> STXT : pour extraire une chaîne de caractères (ici pour extraire un chiffre d'un nombre).

Les élèves obtiennent ainsi un algorithme fonctionnel. Certain réussissent même à le modifier afin de tester tous les nombres à trois chiffres et montrer la conjecture.

Ainsi, il semblerait que l'identification des étapes ne soit pas à la charge des élèves et que l'énoncé propose de travailler directement sur chacune d'entre elles. Si la programmation de ces étapes semble être laissée à la charge des élèves, les interventions de l'enseignant réduisent le travail des élèves.

2.2.3 Travaux menés en seconde (voir Annexe D.4)

Le travail proposé par l'enseignant à cette classe de seconde est constitué de 5 exercices. Chaque exercice a été donné en devoir en temps libre, dans des devoirs non consécutifs, sur une période s'étendant sur 5 mois.

Le premier exercice est une première prise de contact avec l'algorithme, que les élèves doivent exécuter à la main. Cet exercice est suivi de l'établissement d'une conjecture en classe.

Le deuxième exercice présente aux élèves un algorithme échangeant le contenu de deux variables dont ils doivent comprendre les effets. Dans une seconde question, les élèves sont invités à écrire un algorithme renvoyant le plus grand de trois entiers. Les élèves proposent ainsi plusieurs stratégies.

Le troisième exercice demande aux élèves d'élaborer un algorithme permettant de déterminer le chiffre des unités d'un nombre d'un entier quelconque. Cet exercice a posé de nombreuses difficultés aux élèves et l'algorithme final a été élaboré en classe avec l'aide de l'enseignant.

Le quatrième exercice, en faisant référence aux exercices précédents, propose aux élèves d'écrire un algorithme renvoyant les chiffres d'un nombre, puis un algorithme triant trois nombre donnés, avant de construire l'algorithme de Kaprekar.

Ainsi, les exercices sont très dirigés par des énoncés très détaillés, et quand ils le sont moins (exercice 3 par exemple), ils posent de grandes difficultés aux élèves et sont résolus par l'enseignant lors de la correction en classe. De plus, la décomposition en étapes n'est pas à la charge des élèves.

2.3 Tableau récapitulatif des tâches

Dans le tableau suivant, nous mettons en évidence les tâches laissées à la charge des élèves et celles à la charge de l'enseignant dans les différents énoncés et traces de pratiques que nous avons étudiés.

	Math'x	Françoise	Première S	Seconde
Identification des étapes	Donnée par le manuel	Suggérée par les indications de l'enseignante	À la charge de l'enseignant	À la charge de l'enseignant
Décomposition du nombre	L'idée est donnée par le manuel, l'élève n'a qu'à justifier, la fonction MOD est suggérée	Suggérée par les indications de l'enseignante : présentation des fonctions SXT et ENT	À la charge des élèves, mais c'est l'enseignant qui donne l'algorithme final	Très guidée par les énoncés et les corrections
Tri des chiffres	Guidé par l'énoncé, la fonction GRANDE.VALEUR est suggérée	Suggéré par les indication de l'enseignante : présentation de la fonction GRANDE.VALEUR	À la charge de l'élève, mais c'est l'enseignant qui donne l'algorithme final	Guidé par les énoncés
Reconstruction du nombre	À la charge de l'élève	À la charge de l'élève mais présente de nombreuses difficultés	À la charge de l'élève	À la charge de l'élève
Construction de l'algorithme	Pas à la charge de l'élève	À la charge de l'élève	À la charge de l'élève	Très guidée par l'énoncé

Nous voyons ainsi que le manuel est celui qui laisse le moins de tâches à la charge de l'élève. Cependant, dans les séances dirigées par les enseignants, le travail à la charge de l'élève se trouve également réduit par les suggestions ou les corrections données par les enseignants qui limitent la réflexion personnelle de l'élève.

Chapitre 3

Expérimentation

3.1 Conditions de l'expérimentation

L'expérimentation consistera en deux séances en classe entière (24 élèves) de Terminales S, spécialité Mathématiques, en salle informatique. Ces élèves sont familiers avec le logiciel de programmation AlgoBox et ont suivi un enseignement de l'algorithmique dans le cadre du cours de mathématiques depuis la seconde.

Chaque séance durera deux heures, mais dans la seconde, seule la première heure sera consacrée à l'expérimentation décrite ici ; durant la seconde, les élèves travailleront sur la preuve de l'algorithme de Kaprekar qui sera exploitée par Dominique Laval pour ses travaux.

3.2 Scénario des séances en classe

3.2.1 Travail préparatoire à la première séance

Avant la première séance, les élèves auront un premier travail à préparer à la maison afin de découvrir l'algorithme. Il seront pour cela amenés à tester à la main le programme de calcul sur quelques valeurs, à suggérer un test d'arrêt pour la boucle de l'étape 5, et à répertorier les variables nécessaires à l'écriture de l'algorithme correspondant (Voir Annexe A).

3.2.2 Scénario de la première séance

Lors de la première séance, les élèves seront installés par groupes de trois. Les périodes de travail en groupe et de mise en commun au niveau de la classe s'alterneront durant cette séance.

Tout d'abord, il s'agira de corriger les questions posées en travail à la maison.

Les élèves discuteront en groupe des réponses qu'ils ont trouvées à la maison (5-10 min). Un représentant de chaque groupe présentera alors les résultats à l'oral et la synthèse sera faite au tableau par l'enseignant (on ne présentera peut-être que 4 groupes). (10-15 min)

L'établissement de réponses aux questions (voir Annexe A) :

1. Tester à la main ce programme de calcul sur quelques exemples. Émettre une conjecture sur le résultat de cet algorithme.
2. Suggérer un test d'arrêt pour l'étape 5.

est laissé à la charge des élèves ; l'enseignant se contentera de mettre en commun ces réponses et de les soumettre à la validation de la classe, sauf si aucune réponse satisfaisante n'est obtenue au bout de 20 minutes. En ce qui concerne la question 3 (identification des variables à utiliser dans l'algorithme) les différentes réponses pourront être écrites au tableau, mais l'enseignant ne cherchera pas à les valider ou à les infirmer.

Ensuite, l'enseignant incitera les élèves à chercher les différentes étapes à effectuer afin de mettre en œuvre cet algorithme (passage du nombre à la suite de ses chiffres, tri des trois chiffres, passage des chiffres au nombre). La réflexion sera menée en classe entière (20 min).

Une piste envisagée pour mettre en évidence ces étapes sera de modifier les variables didactiques afin de créer un milieu renvoyant de plus importantes rétroactions aux élèves. En effet lors d'une exécution à la main, les étapes sont transparentes et effectuées de manière inconsciente (on passe de 392 à 239 sans y réfléchir). Afin de rendre ces étapes conscientes, l'enseignant suggérera d'appliquer le second point du programme de calcul à un nombre à 10 chiffres. Le passage de 2594131654 à 1123445569 forcera alors les élèves à expliciter leur méthode et à identifier les étapes mises en œuvre.

Ces trois étapes seront alors clairement identifiées par l'enseignant et écrites au tableau, avec un exemple de ce que doit renvoyer l'algorithme pour chacune d'entre elles.

Les élèves travailleront alors en groupes de trois à l'écriture de trois algorithmes distincts permettant d'effectuer chacune de ces trois étapes.

Chaque groupe se consacrera à une étape :

- 3 groupes traiteront le passage du nombre aux chiffres
- 3 groupes traiteront le tri des chiffres
- 2 groupes traiteront la reconstruction du nombre à partir de ses chiffres.

Nous choisissons ce dispositif car il semble que la programmation des 3 étapes par chaque groupe ne soit pas possible dans le temps imparti (séance de 2h). Par ailleurs, ce dispositif permettra un travail collaboratif au sein de la classe.

Les enseignants n'interviendront pas sur l'écriture de l'algorithme sur papier. Toute aide au moment de la programmation sera notée par l'enseignant, et l'algorithme sera enregistré sur clé usb avant la modification par les élèves.

Les élèves passeront de l'algorithme sur papier à la programmation quand ils se sentiront prêts.

Si un groupe a terminé l'algorithme correspondant à l'étape dont il était chargé, il traitera une autre étape, cela jusqu'à ce que tous les groupes aient un algorithme (dans une limite de 45 min de travail).

Les travaux effectués par groupe seront alors mis en commun et validés par la classe (20 min).

3.2.3 Travail préparatoire à la seconde séance

Pour la séance suivante, les élèves devront assembler ces trois étapes afin de construire une première version de l'algorithme de Kaprekar.

3.2.4 Scénario de la seconde séance

La seconde séance sera consacrée à la mise en œuvre de cet algorithme.

Une première version, effectuant le programme de calcul présenté lors de la première séance, sera mise en place à partir des travaux effectués à la maison par les élèves. Ce premier algorithme renverra la liste des étapes effectuées pour arriver à 495 ; par exemple, si on entre le nombre 392, on veut que l'algorithme renvoie :

$$\begin{aligned} 932 - 239 &= 693 \\ 963 - 369 &= 594 \\ 954 - 459 &= 495. \end{aligned}$$

Les élèves seront alors chargés de programmer cet algorithme sur AlgoBox. Certains problèmes sont à prévoir lors de l'exécution de l'algorithme s'ils n'ont pas été identifiés avant ; en effet, si le cas des nombres à trois chiffres égaux n'a pas été exclu, la condition testée dans la boucle sera toujours vérifiée.

On modifiera alors l'algorithme afin qu'il indique combien de passages dans la boucle sont nécessaires pour obtenir 495. Par exemple, si on entre le nombre 392, on veut que l'algorithme renvoie :

Pour 392, on obtient 495 en 3 coups.

Les élèves programmeront ensuite cette version modifiée sur AlgoBox.

La suite de la séance se déroulera sous la direction de Dominique Laval, qui exploitera le travail fait jusque-là pour mener son expérience sur la preuve de l'algorithme de Kaprekar dans le cadre des travaux de recherche menés pour sa thèse.

3.3 Analyse des séances en classe

3.3.1 Déroulement de la première séance

L'installation de la salle et la présentation de la séance aux élèves a duré une dizaine de minutes. Les élèves ont alors été directement répartis en 7 groupes de 3 (il y avait 3 absents). Ils ont eu 5 minutes pour discuter du travail qu'ils avaient préparé à la maison.

À l'issue de ce travail en groupe, deux représentants de deux groupes ont exposé leurs résultats, qui ont ensuite été discutés et complétés par les autres groupes.

Au niveau des résultats, le fait qu'on obtienne toujours 495 ou 0 a été rapidement mis en évidence. Certains sont allés plus loin dans les conjectures en affirmant qu'après le premier passage dans la boucle, on se retrouve dans la suite 792, 693, 594, 495. Les élèves ont remarqué que la somme des chiffres de chaque élément de cette suite était toujours 18, et que « le chiffre du milieu » est toujours 9 (voir Annexe B.1.1).

Le cas de l'apparition du chiffre 0 a été traité de manière unanime sur l'exemple : « Le nombre formé des chiffres de 503 rangés dans l'ordre croissant est 35 ».

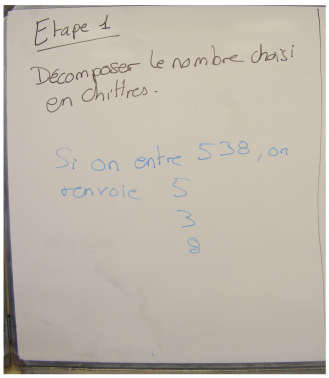
Deux tests d'arrêts ont été proposés (voir Annexe B.1.2) :

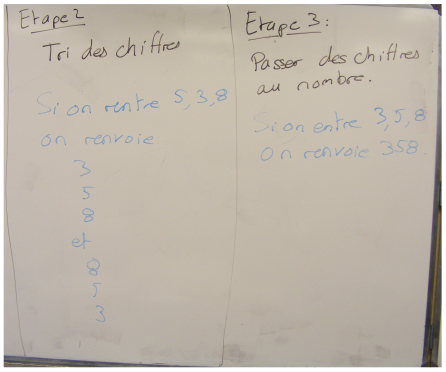
- On s'arrête lorsque la différence de 2 nombres obtenus consécutivement est 0.
- On s'arrête quand on obtient 0 ou 495.

Les variables proposées ont d'abord été un entier A pour stocker le nombre de départ et la différence obtenue à l'étape 4, un entier B pour stocker le nombre obtenu en arrangeant les chiffres dans l'ordre croissant, et un entier C pour stocker le nombre obtenu en arrangeant les chiffres dans l'ordre décroissant. Un autre élève est alors intervenu pour proposer des variables c, d, u pour stocker les chiffres des centaines, dizaines et unités de A (Voir Annexe B.1.3). Un troisième élève s'est alors demandé si l'on pouvait entrer directement ces 3 chiffres plutôt que le nombre A. Un de ses camarades lui a rétorqué qu'il faudrait alors programmer la soustraction ; avant qu'un dernier s'oppose à cela en affirmant qu'il suffirait de construire les nombres B et C à partir de ces chiffres. L'enseignant a alors fait remarqué que l'énoncé précisait « Choisir un nombre à trois chiffres » (Voir Annexe B.1.4).

Ce travail en classe entière a duré 25 minutes.

Dans un second temps, les différentes étapes ont été établies en classe entière. Le travail fait auparavant sur les variables a rendu plus facile ce point, et ce sont les élèves qui ont mis en place ces trois étapes (voir Annexe B.1.5). Elles ont été clairement indiquées au tableau avec un exemple pour chacune d'entre elles :





On a alors demandé aux élèves d'écrire un algorithme correspondant à une étape et de le programmer comme un programme autonome, c'est-à-dire déconnecté des autres étapes. Une élève a alors posé la question de savoir si le premier algorithme (passage du nombre aux chiffres) devait renvoyer les chiffres dans l'ordre ; l'enseignant a répondu que non (Voir annexe B.1.6).

Cette partie a duré 15 minutes.

Les élèves ont alors travaillé par groupes de trois à l'écriture de ces algorithmes. De nombreux groupes ont eu le temps de traiter plusieurs étapes (parfois même les 3). Seul un groupe n'a pas réussi à programmer un algorithme qui fonctionne ; ceci étant principalement dû au fait qu'AlgoBox, comme de nombreux langages, ne comprenne pas les tests de la forme « a<b<c », mais seulement ceux du type « a<b ET b<c ».

De manière générale, la plupart des questions des élèves portaient sur ce qu'AlgoBox était capable de faire ou non (gérer des tableaux, des listes, trouver un maximum, calculer un modulo...).

Cette partie a duré 45 minutes.

Enfin, les algorithmes proposés ont été mis en commun en classe entière.

Pour la première étape, on a obtenu 3 types d'algorithmes (voir Annexe C.1.1) :

- un premier utilisant la fonction partie entière,
- un second utilisant la fonction modulo,
- et un troisième utilisant ces deux fonctions.

Pour la seconde étape, on a obtenu 3 types d'algorithmes (voir Annexe C.1.2) :

- un premier faisant 6 tests successifs pour différencier les 6 cas (c'est celui dont la programmation n'a pas abouti)
- un second effectuant des tests imbriqués et stockant les résultats dans une liste
- un troisième utilisant la structure de liste et les fonctions minimum et maximum d'AlgoBox.

Pour la dernière étape, on a obtenu un seul type d'algorithme du type $100c + 10d + u$ (voir Annexe C.1.3).

Cette partie a duré 15 minutes.

3.3.2 Analyse de la première séance

Des conjectures nombreuses et détaillées

Les conjectures émises par les élèves sont allées au-delà de nos attentes. En effet, nous ne nous attendions pas à ce qu'ils identifient la suite 297, 396, 594, 495. En remarquant comme ils l'ont fait, qu'après une itération de l'algorithme, on obtient un élément de cette suite, les élèves se sont beaucoup rapprochés de la preuve algébrique basée sur le fait qu'après une itération, on obtient un multiple de 99. Par ailleurs, ils ont identifié des particularités aux nombres de cette suite : le chiffre du milieu est toujours 9 et la somme des chiffres est toujours 18. On est ainsi très proche du critère de divisibilité attendu.

Ces conjectures pourront être exploitées par Dominique Laval pour ses travaux concernant la preuve de cet algorithme. Pour notre part, nous constatons que ces conjectures ont entièrement émané des élèves.

Une identification des étapes guidée par l'enseignant mais effectuée par les élèves

Comme le montre précisément la transcription (voir Annexe B.1.5), et contrairement à ce que l'on attendait, l'identification des étapes a émané des élèves.

En effet, si c'est l'enseignant qui suggère la planification de l'algorithme :

> P : Donc, là, la suite, le but de ce qu'on va faire à la suite, c'est de programmer cet algorithme d'accord. Et on va essayer après de programmer sur AlgoBox. Donc, pour pouvoir le programmer, euh, je vais, je vais vous demander, d'après vous quelles sont, quelles vont être les étapes et les petits morceaux de programmes qu'on va avoir besoin de savoir faire, de faire faire à l'ordinateur pour pouvoir programmer cet algorithme.

ce sont les élèves qui commencent à identifier les étapes sans indications supplémentaires :

> E3 : La décomposition du nombre initial choisi en chiffre des centaines, chiffre des dizaines, chiffre des unités.

Ainsi, les pistes envisagées dans l'analyse *a priori* afin de favoriser les élèves à se placer du point de vue de la machine, comme le passage d'un nombre à 10 chiffres, n'ont pas été nécessaires. En fait, la recherche des variables nécessaires à la programmation de l'algorithme a constitué un travail préalable précieux.

En effet, dès ce moment-là, les élèves ont perçu la nécessité d'introduire des variables pour stocker les chiffres :

> E9 : Ben si A c'est une variable où c'est un nombre, pour qu'il puisse le ranger dans l'ordre croissant et décroissant, faut d'abord séparer les chiffres des dizaines, des unités et des centaines. Donc il faut trois variables, un pour le chiffre des unités, un pour le chiffre des dizaines et un pour le chiffre des centaines.

Ainsi, bien que cette étape soit transparente lors de l'exécution manuelle de l'algorithme, la réflexion sur le choix des variables à utiliser a permis aux élèves de la mettre en évidence. Ainsi, lorsque l'élève affirme :

> E3 : L'ordinateur pour arranger les chiffres en ordre décroissant, ça va être important.

il réussit à intégrer le fonctionnement du dispositif.

L'identification de l'étape suivante est également venue des élèves :

> P : Donc étape 1 (écrit au tableau) décomposer le nombre choisi en chiffres. D'accord ? Donc qu'est-ce qu'on va avoir comme étape après ? Une fois qu'on a les chiffres, qu'est-ce qu'on va faire de ces chiffres ? Oui ?
>
> E13 : Il va falloir faire des tests les uns par rapport aux autres, pour voir lequel est le plus petit, lequel est le plus grand. Les comparer...
>
> P : Donc des tests par rapport aux autres pour savoir lequel est le plus petit, lequel est le plus grand. De manière générale, qu'est-ce qu'on va vouloir faire sur ces chiffres ? Oui ?
>
> E1 : Les classer.
>
> P : Oui donc ?
>
> E1 : Les classer dans l'ordre.

Enfin, l'identification de la dernière étape est plus laborieuse, probablement car elle est sous-entendue par les élèves :

> Et qu'est-ce qu'il va nous rester à faire après ?
>
> E13 : La différence.
>
> P : La différence de quoi ?
>
> E11 : Reconstituer des nombres.
>
> P : Alors, avant de faire la différence, effectivement, il va falloir reconstituer les nombres. Oui ?
>
> E10 : Euh, oui, pour la différence, des nombres triés en ordre décroissant par celui trié en ordre croissant.

Ici, les élèves peinent à identifier l'étape. En effet, lors de l'exécution manuelle, le problème de la reconstitution ne se pose pas, il est propre au fonctionnement du dispositif informatique. Cependant, les élèves finissent par mettre en évidence la reconstitution en énonçant directement l'algorithme à appliquer :

> E11 : Ben on fait fois 100, fois 10...

Au final, les trois étapes ont été identifiées par les élèves.

La programmation des étapes entièrement à la charge des élèves

La programmation des étapes a été entièrement laissée à la charge des élèves. L'enseignant n'est intervenu que sur des questions ponctuelles concernant le fonctionnement d'AlgoBox.

Cette partie de la séance a permis aux élèves de construire des schémas de programme pour chacune des étapes mises en évidence, afin de pouvoir les réinvestir lors de la conception de l'algorithme.

Cependant, tous les groupes n'ayant pas avancé à la même vitesse, ils ne se sont pas tous familiarisés de la même façon avec ces schémas. La mise en commun au tableau des trois étapes avait pour objectif de permettre à tous les élèves d'intégrer un schéma pour chaque étape.

Des questions sur le fonctionnement du dispositif

Durant la programmation des étapes par groupe de trois, l'essentiel des questions posées par les élèves concernaient le fonctionnement d'AlgoBox, et en particulier, savoir si telle ou telle fonction existait déjà ou non dans ce logiciel de programmation (voir Annexe B.1.7).

En effet, en tant que débutant, les élèves ne connaissaient pas toutes ces fonctions. Au delà de préoccupations purement techniques, ces questions révèlent également une nécessité d'avoir déjà intégré des schémas à objectifs plus petits. En effet, pour programmer la décomposition du nombre en chiffres, des micro-schémas du type "extraire une partie entière" ou "calculer le reste dans une division euclidienne" doivent pouvoir être utilisés par les élèves. Ainsi, les élèves se posent la question de savoir si AlgoBox peut effectuer ces calculs grâce à des fonctions pré-programmées, ou s'ils vont devoir programmer ces sous-fonctions eux-mêmes.

De même, le groupe ayant programmé le tri des chiffres à l'aide des fonctions MIN et MAX (voir Annexe C.1.2) a intégré des schémas faisant appel à des fonctions d'AlgoBox pour calculer le minimum ou le maximum d'une liste, mais ne s'est pas approprié de schéma permettant de calculer la médiane d'une liste, pour laquelle une fonction existe pourtant dans AlgoBox, et qui leur aurait permis d'identifier "le chiffre du milieu". À la place, ils ont programmé une fonction leur permettant d'identifier ce troisième chiffre. Ils se sont ainsi familiarisés avec un schéma de programmation d'identification d'un nombre différent de deux autres.

Les limites de nos hypothèses

Dans notre première hypothèse, nous assimilions l'algorithmique et la programmation. Le travail effectué par un groupe de trois élèves autour d'un algorithme de tri montre les limites de cette hypothèse. En effet, ces élèves avaient écrit un algorithme tout à fait correct, utilisant 6 tests successifs :

Algorithme proposé :

On entre 3 variables a, b, c, g, h, i.

✓ Si $a \leq b \leq c$
alors $a \to g$, $b \to h$, $c \to i$

✓ Si $a \leq c \leq b$
alors $a \to g$, $c \to h$, $b \to i$

✓ Si $b \leq a \leq c$
alors $b \to g$, $a \to h$, $c \to i$

✓ Si $b \leq c \leq a$
alors $b \to g$, $c \to h$, $a \to i$

✓ Si $c \leq a \leq b$
alors $c \to g$, $a \to h$, $b \to i$

Si $c \leq b \leq a$
alors $c \to a$, $b \to b$, $a \to c$

Afficher a, b et c
Afficher c, b et a

Impliqués

3. Programmer cet algorithme sur Algobox.

Cependant, programmé tel quel en AlgoBox, l'algorithme ne fonctionne pas comme attendu. En effet, AlgoBox n'autorise que des comparaisons entre deux éléments. Ainsi, au lieu d'écrire A⩽B⩽C, il faut écrire A⩽B ET B⩽C.

Notre seconde hypothèse suggérait que les élèves ne rencontraient pas de difficultés liées au langage. Bien que cela ce soit essentiellement avéré, ils ont tout de même été confrontés à quelques subtilités les empêchant d'agir en complète autonomie.

Étudions plus précisément le moment où un élève a proposé d'effectuer un traitement sur les chiffres plutôt que sur le nombre :

> E3 : Dans ce cas ce serait peut-être plus utile de voir dans l'entrée les chiffres, lui dire, on rentre le chiffre des centaines, on rentre le chiffre des dizaines, on rentre le chiffre des unités. Ça empêcherait de faire une décomposition à la machine.

Un autre élève identifie alors rapidement le problème que cette vision entraînerait au niveau du traitement de la soustraction :

> E8 : Mais après pour le traitement de la soustraction, ça va être difficile.

Un troisième propose alors de reconstituer le nombre :

> E11 : On peut choisir le chiffre c fois 100 plus d fois 10 plus u.
> E8 : Oui, oui, c'est faisable.

Nous avons choisi de ne pas développer cette stratégie pendant l'expérimentation, et l'enseignant a réorienté les élèves vers un traitement des nombres.

Cependant, la "solution" proposée par E11 était inappropriée. En effet, cette étape est imbriquée dans une boucle, il est donc nécessaire que les variables qu'elle renvoie soient du même type en entrée qu'en sortie. Si on reconstitue le nombre pour effectuer la soustraction, il faudra donc décomposer le résultat afin de le renvoyer dans la boucle.

Cet exemple illustre donc une conception inadéquate des structures répétitives.

3.3.3 Déroulement de la seconde séance

La seconde séance s'est déroulée dans des conditions un peu particulières. En effet, les élèves fêtaient Carnaval ce jour-là et étaient donc un peu moins concentrés que d'habitude. Par ailleurs, contrairement à ce qui était prévu, cette séance s'est déroulée un mois après la première (conseil de classe et vacances empêchant deux séances consécutives).

La mise en commun par groupe du travail effectué à la maison (algorithme sur papier) a duré environ 10 minutes.

L'enseignant a ensuite mis en commun au tableau les travaux des élèves. Le choix des algorithmes pour chaque étape a été laissé à la discrétion de chaque groupe d'élève. Un problème s'est posé au niveau du test d'arrêt de la boucle TantQue : il y a eu une discussion entre les élèves pour savoir si la condition de continuation était ($A \neq 495$ ET $A \neq 0$) ou ($A \neq 495$ OU $A \neq 0$). L'enseignant a choisi de laisser cette question ouverte pour la programmation (Voir Annexe B.2.1). Cette partie a duré une quinzaine de minutes.

Les élèves ont ensuite été invités à programmer cet algorithme sur AlgoBox. Malgré les difficultés rencontrées, 5 des 7 groupes ont réussi à obtenir un algorithme fonctionnel après 20 minutes.

Après une rapide mise au point en classe entière, où les élèves ont notamment tranché la question de la condition de continuation grâce aux rétroactions

fournies par la programmation de l'algorithme, les élèves sont passés à la programmation du second algorithme renvoyant le nombre d'itérations nécessaires pour obtenir 495. Cette phase a duré environ 40 minutes.

Les programmes écrits par les élèves ont ensuite été projetés au tableau devant la classe entière.

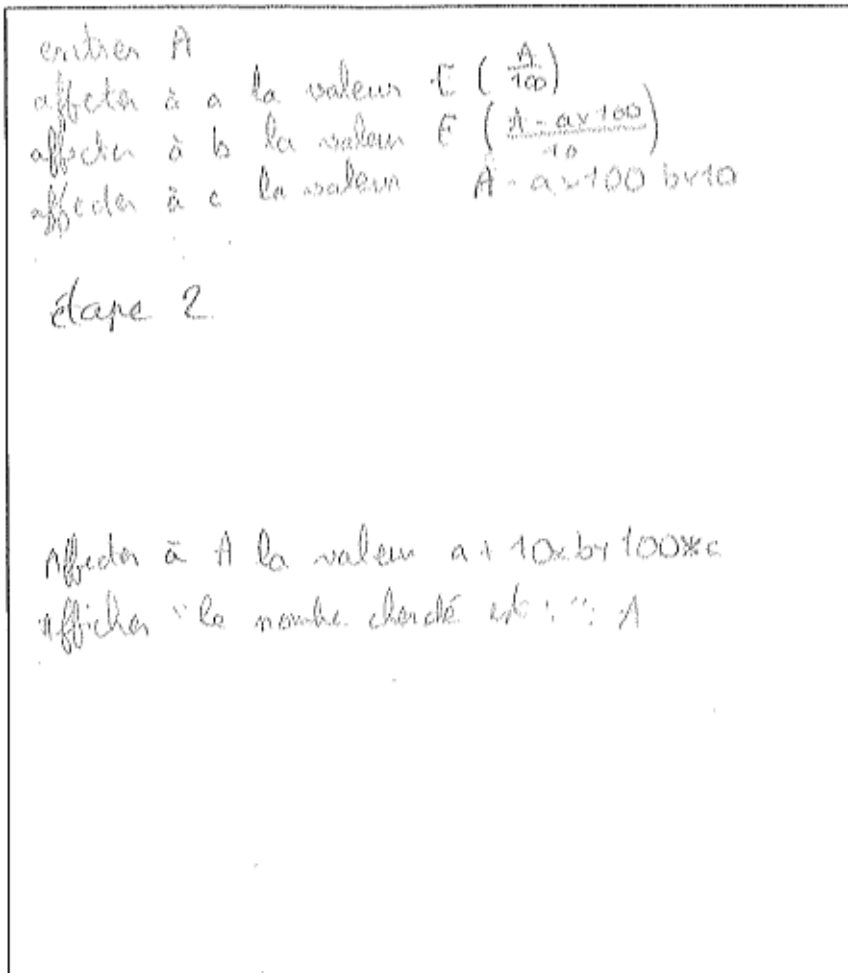
3.3.4 Analyse de la seconde séance

La reconstruction de l'algorithme entièrement à la charge des élèves

L'écriture de l'algorithme a été faite à la maison par les élèves, et était donc entièrement à leur charge. De manière générale, les élèves n'ont pas rencontré beaucoup de difficultés pour relier les étapes qui avaient été établies lors de la séance précédente.

Il faut tout de même noter que les groupes qui n'avaient pas traité toutes les étapes, et en particulier ceux qui n'avaient pas programmé la seconde étape, ont rencontré plus de difficultés que les autres. En effet, la mise en commun au tableau à la fin de la première séance avait été assez rapide et probablement insuffisante pour que les groupes n'ayant pas du tout traité la seconde étape puissent se l'approprier. On suppose que ce phénomène se présente uniquement pour les groupes n'ayant pas traité l'étape 2 car il semble que c'est la plus difficile à programmer pour les élèves. En effet, quelle que soit la stratégie utilisée, cette étape fait intervenir un grand nombre de variables et nécessite l'emploi soit de plusieurs tests, soit de fonctions sur les listes.

Ainsi, s'ils comprennent la méthode descendante indiquée pour la conception de cet algorithme, ils ne parviennent pas à l'appliquer car ils n'ont pas intégré les schémas auxquels cette méthode fait appel : l'encapsulation de tels schémas n'est ainsi pas triviale pour les débutants.



On voit sur cet extrait de copie que l'élève a compris qu'il fallait enchaîner les trois étapes, et s'il insérait un algorithme de tri correct en lieu et place de *étape 2*, son algorithme serait fonctionnel.

Ainsi, il semblerait que le réinvestissement d'un schéma de programme dans le cadre de la planification d'un algorithme complexe nécessite une bonne intégration préalable de ce schéma par l'élève. Cette appropriation semble être favorisée par la programmation de ce schéma par l'élève lui-même, puisque dès lors qu'une étape avait été programmée par un élève lors de la première séance, l'élève a su la réinvestir pour la seconde séance.

Des difficultés liées au langage

De nombreuses difficultés se sont présentées à la classe lors de l'expression de la condition de continuation de la boucle.

En effet, s'il était clair que le test d'arrêt adéquat était ($A = 0$ OU $A = 495$), le passage à la condition de continuation n'a pas été trivial, ni pour les élèves, ni pour l'enseignant (voir Annexe B.2.1).

Ainsi, si l'algorithme exprimé en langage naturel à l'aide d'une boucle du type CONTINUER JUSQU'À A=0 OU A=495 a bien été établi, le passage au langage AlgoBox et à l'utilisation d'une boucle TANTQUE a posé de nombreuses

difficultés.

En effet, d'après les recherches menées par Rogalski et Samurçay [22], les débutants sont moins à l'aise avec des boucles du type "test/traitement" qu'avec celles du type "traitement/test" qui correspondent à la forme spontanée de structure itérative :

> Novices' "spontaneous" iterative models have the following structure : description of action, repetition counter, repetition mark and expression of the end control. Anticipation and explicitation of the end control are not spontaneously aviable probably this control is implicit in ordinary action plans. these features make it more difficult to deal with "test/process" plans (e.g. the WHILE loop in Pascal). Difficulties stem from representing and expressing a condition about an object on wich they have not yet operated. [22]

Outre cette prédisposition aux structures répétitives du type RÉPÈTE... JUSQU'À..., c'est aussi l'inversion du test d'arrêt qui pose problème, et Soloway [28] constate que la structure TANTQUE pose encore des problèmes aux étudiants "avancés".

3.3.5 Tâches à la charge de l'élève, tâches à la charge de l'enseignant

Nous menons ici une analyse des tâches comparable à celle menée en 2.3.

Identification des étapes	Décomposition du nombre	Tri des chiffres	Reconstruction du nombre	Construction de l'algorithme
L'idée de la décomposition en étapes et suggérée par l'enseignant, mais les étapes sont identifiées par les élèves.	Entièrement à la charge de l'élève.	Entièrement à la charge de l'élève.	Entièrement à la charge de l'élève	Entièrement à la charge de l'élève.

Nous constatons que la part de travail à la charge de l'élève est très importante, et que dès lors que l'enseignant suggère une décomposition de l'algorithme en étapes, les élèves travaillent en quasi-autonomie.

Chapitre 4

Conclusion

4.1 Les principaux résultats de nos recherches

Dans nos recherches, nous avons voulu examiner les effets de la planification sur la conception d'algorithmes complexes par des élèves de lycée. Nous nous sommes basés sur des recherches menées dans les années 80 par Rogalski et Samurçay. Nous avons ensuite choisi de mettre en place une expérimentation autour d'un algorithme particulier.

Nous avons choisi de travailler autour de l'algorithme de Kaprekar pour plusieurs raisons :

- La description de cet algorithme est simple et son exécution *à la main* est largement à la portée des élèves de lycée,
- Mais son exécution *sur machine* présente une certaine complexité puisqu'elle fait intervenir une boucle et plusieurs étapes à l'intérieur de celle-ci. Cette complexité justifie l'utilisation de la planification pour la programmation de cet algorithme.

Dans notre expérimentation, nous avons considéré les élèves de lycée comme des débutants en programmation déjà alphabétisés. Notre objectif était de leur faire concevoir un algorithme complexe, tâche qu'ils n'ont pas l'habitude de rencontrer.

Afin de mieux analyser les résultats de notre expérimentation, nous avons également choisi d'étudier des expériences faites par d'autres enseignants. Nous avions pour but de comparer les tâches à la charge des élèves et celles à la charge des enseignants dans les différentes expériences étudiées. Il est apparu que notre expérimentation dans laquelle les élèves sont guidés par la planification semble laisser plus de place au travail de l'élève que dans les autres expériences où l'enseignant prend une part du travail à sa charge.

4.1.1 La TSD : Un outil pertinent pour observer l'autonomie des élèves développée par la planification

Le but de notre expérimentation était d'observer le travail des élèves lors de la planification d'algorithmes complexes. Nous avons pu observer l'autonomie dont ils faisaient preuve en nous plaçant dans le cadre théorique de la TSD.

Il nous semble que le fonctionnement de la TSD est du à une situation prévue pour accorder le plus de responsabilités aux élèves grâce à une approche "planification". L'analyse a priori de cette situation nous a en effet menés à

établir un découpage précis en phases de travail, afin de distinguer clairement les responsabilités de l'élève et celles de l'enseignant. C'est cette analyse qui nous a permis de dévoluer la situation aux élèves. En effet, l'étude dans notre analyse a priori d'une planification de l'algorithme nous a permis d'identifier des étapes précises et d'étudier l'a-didacticité de chacune des phases de travail nécessaires à la mise en place de ces étapes de planification.

Ainsi, nous avions prévus que l'identification des étapes reste à la charge de l'enseignant. Mais les questions à préparer avant la séance, et notamment celle demandant aux élèves d'identifier les variables nécessaires à l'écriture de l'algorithme, ont produit un milieu suffisamment riche pour que les élèves évoluent en a-didacticité dans cette phase de travail. Par ailleurs, les élèves se sont également retrouvés dans une situation de formulation et de validation puisqu'ils ont du expliquer leurs idées à toute la classe.

La seconde phase de travail identifiée grâce au point de vue de la planification est celle de la programmation des étapes de l'algorithme. Là encore, le milieu, constitué de la précédente discussion en classe et des rétroactions fournies par le dispositif informatique lors de la programmation, mais aussi les connaissances anciennes des élèves (en programmation et en mathématiques) ont permis, conformément à l'analyse a priori, de dévoluer cette phase de travail aux élèves.

La dernière phase de reconstruction de l'algorithme s'est vue au maximum dévoluée aux élèves, notamment grâce aux rétroactions fournies par le dispositif informatique. En effet, ces rétroactions leur ont non seulement permis de corriger de manière autonome (sans intervention de l'enseignant) des erreurs de syntaxe, mais également de répondre à des questions théoriques. Ainsi, le problème de savoir si le test de continuation de la boucle devait être ($A \neq 495$ OU $A \neq 0$) ou ($A \neq 495$ ET $A \neq 0$) (voir 3.3.4) a été tranché grâce aux rétroactions fournies par le programme. En effet, le test (TANT QUE $A \neq 495$ OU $A \neq 0$) produisait une boucle infinie renvoyant une infinité de 495 ou de 0. Les élèves ont ainsi compris que ce test était erroné et ont cherché à savoir pourquoi. Ils se sont ainsi rendu compte que tout nombre est différent de 495 ou différent de 0.

Ainsi, les tâches laissées à la charge des élèves lors de l'expérimentation ont même été plus importantes que prévu dans l'analyse a priori menée du point de vue de la planification. Par exemple, l'identification des étapes devait rester à la charge de l'enseignant, mais dès lors qu'il a suggéré l'utilisation d'une planification en étapes, les élèves les ont eux-mêmes identifiées très rapidement.

Les approches par la Théorie des Situations, développée en didactique des mathématiques, et par la planification, étudiée en psychologie de la programmation, semblent donc complémentaires pour étudier ce type de situations d'enseignement.

4.1.2 Les apports de la planification

La planification a permis aux élèves de gagner en autonomie lors de la conception de l'algorithme. En effet, dès lors que cette démarche a été suggérée par l'enseignant, le reste du travail (identification des étapes, programmation des étapes et programmation de l'algorithme) a été mené par les élèves.

Ainsi, cette décomposition en étapes de traitement semble particulièrement pertinente pour le problème étudié.

L'acquisition des schémas de programmation s'est également déroulée en a-didacticité, puisque les élèves ont programmé ces schémas en utilisant les rétroactions fournies par le dispositif informatique.

Le réinvestissement de ces schémas dans la programmation de l'algorithme n'a alors pas posé de problème lorsqu'ils avaient été correctement encapsulés. Notre expérimentation a mis en évidence une autonomie accrue des élèves lors de la conception d'algorithmes complexes dès lors qu'ils ont bien intégré les schémas de programmes sous-jacents. Ainsi, il semble nécessaire que les élèves se soient bien familiarisés avec ces schémas afin de pouvoir les réinvestir dans la programmation d'un autre algorithme. Il apparaît aussi que la lecture d'un algorithme ne suffise pas à l'intégrer comme schéma de programme, mais que le fait de programmer soi-même cet algorithme permette de mieux se l'approprier.

4.1.3 Mise en place d'une méthode de planification semi-descendante

Nous avons initié les élèves à une méthode consistant à identifier une planification générale de l'algorithme avant de programmer des étapes comme des sous-programmes, afin de pouvoir les intégrer dans la programmation finale de l'algorithme.

Cette méthode est ainsi *semi-descendante*. En effet, les élèves ont d'abord été amenés à considérer une planification générale de l'algorithme, avant d'étudier des schémas de programmes correspondant aux étapes mises en œuvre dans l'algorithme complexe, puis ils ont assemblé ces schémas afin de programmer l'algorithme lui-même.

Il semble que cette méthode permette de laisser une part de tâches à la charge de l'élève plus importante que les méthodes abordées dans les manuels ou dans les pratiques rapportées par d'autres enseignants.

4.2 Limites et perspectives

4.2.1 Les limites de nos hypothèses

Si la planification peut sembler efficace, elle présente tout de même des limites lorsqu'elle est utilisée par des débutants. En effet, nos hypothèses d'alphabétisation des élèves ont été mises en défaut. Des difficultés liées au langage ont ainsi empêché les élèves de profiter pleinement des bénéfices de la méthode employée.

Par ailleurs, si nous avons utiliser le cadre de la TSD pour étudier les effets de la planification d'algorithmes complexes, il est vrai que nous ne nous sommes pas posé la question de l'institutionnalisation des savoirs appris. On peut ainsi se demander par quels moyens l'enseignant peut institutionnaliser les connaissances relatives à la numération nécessaires à la décomposition du nombre en chiffres par exemple ; on peut penser que c'est peut-être possible lors de

la preuve de l'algorithme (voir travaux de Dominique Laval). Pour ce qui est des connaissances relatives à la planification de programmes elle-même, comment les institutionnaliser ? Faut-il avoir recours à des méthodes ?

4.2.2 Une méthode dont l'efficacité n'a pas été testée

Des méthodes de programmation ont été mises au point vers la fin des années soixante pour améliorer l'efficacité des programmeurs professionnels. Les didacticiens se sont intéressés à l'enseignement de telles méthodes à des novices :

> une programmation méthodique dès les premiers apprentissages évite de prendre des habitudes de programmation empirique et désordonnée, tout en facilitant la résolution des problèmes de programmation. [23]

Ainsi, ces méthodes ont pour but de faciliter la résolution de certaines classes de problèmes :

> Une méthode peut être considérée comme un outil d'aide à la résolution de problèmes : elle ne décrit pas les procédures à utiliser, mais guide l'utilisateur pour la recherche de stratégies de résolution de problèmes d'une certaine classe. Une méthode exprime un processus commun de recherche de modes de résolution, processus valable pour une classe de situations qui ont un ensemble de points communs.[21]

Dans nos recherches, nous avons suggéré une méthode de programmation à nos élèves, mais nous ne leur avons pas présentée comme générique pour une certaine classe de problèmes. Il n'est pas sûr que les élèves aient recours à cette méthode pour résoudre un problème similaire à celui traité dans l'expérimentation.

4.2.3 La planification : un point aveugle dans le curriculum

Le programme de lycée introduisant des notions d'algorithmique insiste sur l'enseignement de structures de programmation, et notamment sur les boucles, mais ne fait pas mention de planification de programmes ou d'algorithmes. Cependant, il semble nécessaire d'introduire la planification dès lors qu'on demande aux élèves de programmer de manière autonome des algorithmes complexes (c'est-à-dire comportant plusieurs étapes de traitement), comme le montre notre étude comparative des expériences menées par les enseignants.

Ainsi, les algorithmes que les élèves sont amenés à programmer au lycée semblent être cantonnés à des algorithmes simple, en excluant l'imbrication de structures dans des boucles par exemple. Les algorithmes de tri sont par exemple hors du programme.

> Dans le cadre de cette activité algorithmique, les élèves sont entraînés à :
> - décrire certains algorithmes en langage naturel ou dans un langage symbolique ;
> - en réaliser quelques-uns à l'aide d'un tableur ou d'un programme sur calculatrice ou avec un logiciel adapté ;
> - interpréter des algorithmes plus complexes.
>
> [3]

Les "algorithmes plus complexes" ne sont pas destinés à être programmés, mais uniquement "interprétés".

Par ailleurs, l'enseignement d'une démarche de planification requiert de faire concevoir aux élèves des algorithmes complexes. En effet, il semble peu approprié de faire identifier les étapes d'un algorithme qui n'en comporte qu'une. Cependant, l'utilisation de boucles ou de tests peut justifier à elle seule le recours à la

planification, puisque les étapes de traitements intervenant à l'intérieur de ces structures doivent être clairement identifiées par les élèves.

4.2.4 Vers d'autres recherches

Notre recherche semble avoir mis en évidence l'efficacité de l'introduction d'une méthode semi-descendante pour la programmation d'algorithmes complexes par des débutants alphabétisés. Cependant, nous n'avons pas étudié l'institutionnalisation d'une telle méthode. En effet, cette méthode a été proposée aux élèves mais n'a pas été identifiée. On peut se demander quels seraient les effets de l'enseignement d'une telle méthode et de son utilisation en autonomie (sans guidage ni suggestion) par les élèves.

Il serait peut-être intéressant de mener de telles recherches dans l'enseignement supérieur afin d'éviter les problèmes dus à l'alphabétisation incomplète des élèves rencontrés dans notre étude.

Par ailleurs, il serait intéressant de se pencher sur un moyen d'intégrer des schémas de programmes pour des débutants. En effet, nous avons vu que cette encapsulation était nécessaire pour que les élèves puissent réinvestir ces schémas lors de la programmation d'algorithmes complexes, mais que certains n'avaient pas réussi à effectuer complètement cette encapsulation.

D'autre part, nous avons ici articulé la planification de programmes, notion tirée de la psychologie de la programmation, avec la théorie des situations didactiques afin d'étudier la conception d'algorithmes complexes par des élèves de lycée. Il serait intéressant de poursuivre cette recherche avec d'autres théories, à la fois du côté didactique des mathématiques et étude de la programmation.

En effet, l'évolution du curriculum introduisant l'algorithmique dans les programmes de lycée traduit une influence réciproque des mathématiques et de l'informatique. Si l'informatique puise ses questions dans des problèmes mathématiques, les performances permises par les avancées informatiques ont également transformé le paysage mathématique. L'enseignement des mathématiques semble donc être aujourd'hui profondément lié à celui de l'informatique.

Cette nouvelle donne pose de nouvelles questions didactiques. Jusqu'ici, les didactiques des mathématiques et de l'informatique se sont développées en parallèle, et la situation actuelle nécessite de croiser ces deux points de vue afin de développer des outils adaptés à l'étude de l'enseignement de l'algorithmique au lycée.

Bibliographie

[1] Ressources pour la classe de seconde, 2009.

[2] Seconde épreuve d'admissibilité du concours de recrutement des professeurs des écoles, 2010.

[3] Programme de l'enseignement spécifique et de spécialité de mathématiques, 2011.

[4] Guy Brousseau. *Théorie des situations didactiques : didactiques des mathématiques 1970-1990.* Recherches en didactique des mathématiques. La pensée sauvage, Grenoble, 1998. Réimpression : 2004.

[5] IREM de Franche-Comté. Kaprekar. `http://www-irem.univ-fcomte.fr/index.php?id=article_13469_65_1&itemracine=9884&global_id_item=9884`.

[6] Académie de Poitiers. L'algorithme de kaprekar en algobox.

[7] R. Douady. Jeux de cadres et dialectique outil-objet. *Recherches en didactique des mathématiques*, 1986.

[8] R. Duval. Transformations de représentations sémiotiques et démarches de pensée en mathématiques. *Actes du XXXIIe Colloque COPIRELEM.*

[9] J.M. Hoc. Psychologie cognitive de la planification. *Editions PUG*, 1987.

[10] J.M. Hoc. Prise de conscience et planification. *Psychologie française*, 1988.

[11] E. Horowitz and S. Sahni. *Fondamentals of Computer Algorithms.* Edition Pitman, 1978.

[12] D-E Knuth. *Algorithmes fondamentaux, l'art de la programmation d'ordinateur.* Addison-Wesley Publishing Company, 1968.

[13] J. Kuntzmann. *Méthode numérique.* Collection Enseignement des sciences. Edition Hermann, 1969.

[14] J.-B. Lagrange. Représentations mentales des données informatiques et difficultés d'acquisition chez des débutants en programmation. *Le bulletin de l'EPI*, 67-68, 1992.

[15] MATh.en.JEANS. L'algorithme de kaprekar, 1991.

[16] C. T. Nguyen. *Étude didactique de l'introduction d'éléments d'algorithmique et de programmation dans l'enseignement mathématique secondaire à l'aide de la calculatrice.* PhD thesis, 2005.

[17] C. T. Nguyen and A. Bessot. Introduire des éléments d'algorithmique et de programmation dans l'enseignement secondaire ? une ingénierie didactique. *Petit x*, 83, 2010.

[18] C. T. Nguyen and A. Birebent. Conception d'une ingénierie didactique pour l'introduction d'éléments d'algorithmique et de programmation dans l'enseignement secondaire des mathématiques. *Les cahiers de Leibniz*, 125, 2005.

[19] J.-C. Rauscher. Éléments pour la formation initiale des professeurs d'écoles à partir de l'algorithme de kaprekar. *Actes du XXXIVe Colloque COPIRELEM*.

[20] J. Rogalski. Alphabétisation informatique, problèmes conceptuels et didactique. *Bulletin de l'APMEP*, 347, 1985.

[21] J. Rogalski. Enseignement de méthodes de programmation dans l'initiation à l'informatique. *Le bulletin de l'EPI*, 1988.

[22] J. Rogalski and R. Samurçay. Aquisition of programming knowledge and skills. *Psychology of programming*, 1990.

[23] J. Rogalski, R. Samurçay, and J.M. Hoc. L'apprentissage des méthodes de programmation comme méthode de résolution de problème. *Le Travail Humain*, 1988.

[24] R. Samurçay. De faire à faire faire. planification d'actions dans la situation de programmation. *Enfance*, 1985.

[25] R. Samurçay. Signification et fonctionnement du concept de variable informatique chez des élèves débutants. *Educational Studies in Mathematics*, 16 :144–161, 1985.

[26] R. Samurçay. Plans, et schémas de programme. *Psychologie française*, 1988.

[27] R. Samurçay and A. Rouchier. Apprentissage de l'écriture et de l'interprétation des procédures récursives. *Recherches en didactique des mathématiques*, 10/2-3, 1991.

[28] E. Soloway, K. Ehrlich, J. Bonar, and J. Gresspan. What do novices know about programming. *Yale University*, 1982.

[29] G. Vergnaud. A classification of cognitive tasks and operations of thought involved in addition and subtraction problems. *Addition and Subtraction : A Cognitive Perspective*, 1982.

Annexes

Chapitre A

Document élève

L'algorithme de Kaprekar

1 Découverte de l'algorithme

À faire pour le 21/02

On considère le programme de calcul suivant :

1. Choisir un nombre entier de trois chiffres.
2. Former le nombre obtenu en arrangeant les chiffres du nombre choisi en 1. dans l'ordre croissant.
3. Former le nombre obtenu en arrangeant les chiffres du nombre choisi en 1. dans l'ordre décroissant.
4. Calculer la différence des nombres obtenus en 2. et 3.
5. Recommencer (à partir de l'instruction 2.) avec le résultat obtenu en 4, jusqu'à obtenir un nombre déjà obtenu.

1. Tester à la main ce programme de calcul sur quelques exemples. Émettre une conjecture sur le résultat de cet algorithme.

2. Suggérer un test d'arrêt pour l'étape 5.

On envisage d'écrire l'algorithme correspondant à ce programme de calcul.

3. Déterminer les variables nécessaires à l'écriture de cet algorithme.

2 Les étapes de l'algorithme

Séance du 21/02

1. Quelles sont les étapes à programmer afin de pouvoir effectuer cet algorithme ?

2. Proposer un algorithme pour l'étape traitée par votre groupe :
   Étape traitée :

Algorithme proposé :

3. Programmer cet algorithme sur Algobox.

3 La construction de l'algorithme de Kaprekar

À faire pour le 21 mars

1. Construire un algorithme effectuant le programme de calcul présenté dans la partie 1. Par exemple, si on entre le nombre 392, on veut que l'algorithme renvoie :

$$\begin{aligned} 932 - 239 &= 693 \\ 963 - 369 &= 594 \\ 954 - 459 &= 495 \end{aligned}$$

Séance du 21/03

2. Programmer cet algorithme sur Algobox.
3. Modifier l'algorithme précédent de sorte qu'il indique combien de passages dans la boucle sont nécessaires pour obtenir 495. Par exemple, si on entre le nombre 392, on veut que l'algorithme renvoie : Pour 392, on obtient 495 en 3 coups.

4. Programmer l'algorithme modifié sur Algobox.

Chapitre B

Transcriptions

B.1 Première séance

Dans toutes nos transcriptions, P désigne l'enseignant, O désigne l'observateur et E_i désigne un élève.

B.1.1 Conjectures

E1 : On a toujours trouvé à chaque fois 495 à la fin de l'algorithme.

P : D'accord, donc ça c'est la réponse à la première question, c'est ça ?

E1 : Euh oui.

P : C'est ce qu'on vous demande de faire, OK, on trouve toujours 495, c'est ce que t'as remarqué, c'est ça ?

E1 : Euh oui.

P : Ça c'est la conjecture que vous émettez. OK.

[...]

P : Donc pour la question 1, ce que vous nous proposez, euh, c'est, donc la conjecture que vous faites à la question 1, c'est on obtient 495 tout le temps, c'est ça ?

E1 : Oui.

P : On obtient toujours 495. OK ?

[...]

P : Ok, donc, est-ce qu'un autre groupe a à me proposer des réponses différentes pour ces questions ? Donc, euh, vas-y.

E2 : Donc pour la question 1 on peut trouver 0 aussi.

P : Alors, on peut trouver 0 pour la question 1. Dans quels cas est-ce que tu as trouvé 0 par exemple ?

E2 : Si les trois chiffres sont les mêmes.

P : Si les trois chiffres sont les mêmes. (À élève 1) Est-ce que tu es d'accord avec ça ?

E1 : Euh, oui.

P : Oui, d'accord. Donc toi tu m'as dit on obtient toujours 495, mais en fait, c'est pas toujours. Parce que pourquoi est-ce qu'on obtient 0 si les trois chiffres sont les mêmes ?

E2 : Parce que que ce soit dans l'ordre croissant ou dans l'ordre décroissant qu'on les range, ça sera le même...

P : Ça sera le même nombre.

E2 : . . . donc ça fera le même chiffre moins le même chiffre ça fera 0.

P : D'accord ? Vous êtes d'accords avec ça ? Donc on obtient toujours, pas toujours 495, on obtient 495 ou 0.

[. . .]

P : Est-ce qu'il y a des gens qui ont trouvé d'autres choses sur les conjectures ici ? Qui ont conjecturé d'autres choses que juste on arrive toujours à 495 ? Oui ?

E3 : Ça arrive qu'on trouve des fois avant 693.

P : Donc ça, ce que tu me dis c'est que avant 495, le nombre qu'on trouve, euh, quand on fait un passage de moins dans la boucle, c'est 693.

E3 : Pour tout nombre constitué de trois chiffres dont deux chiffres sont différents. Pour pas que ça fasse 0.

P : Oui. D'accord. Donc, déjà, on est bien d'accord, quels sont les nombres qui donnent 0, c'est ce qui sont constitués de trois chiffres identiques et est-ce que c'est seulement eux, qui donnent 0 ? C'est des conjectures pour l'instant. D'accord ? On vous demande pas de le montrer. Est-ce que vous avez trouvé d'autres nombres que des nombres qui ont les trois chiffres identiques qui donnent 0 ? Non, donc, ceux qui donnent 0, ceux-là c'est si on a trois chiffres identiques. Donc c'est un cas un petit peu à part. Donc ce que tu me disais, ce que tu me disais c'est que l'étape d'avant 495, on va obtenir 693, c'est ça ? Donc on obtient 495 ou 0, et avant 495, on obtient 693. Tu peux me donner un exemple juste pour euh. . . Oui ?

E4 : Ben moi justement, j'ai un contre-exemple à donner.

P : Alors, on va lui demander un exemple et après je te demanderai ton contre-exemple. Alors donne-moi un exemple.

E3 : Euh. . . J'ai pris 754.

P : (écrivant) 754, ok, donc ça nous donne quoi ?

E3 : 754 moins 457 ça fait 293.

P : (écrivant) moins 457 ça fait 293 c'est ça ?

E3 : Oui.

P : OK.

E3 : Euh 97 pardon.

P : (écrivant) 97. Ensuite on a . . .

E3 : On a 792 moins 279, 693.

P : Humhum.

E3 : 963 moins 369, 594, ça s'arrête là.

P : Alors, ça s'arrête pas là puisqu'on a dit que nous on s'arrêtait à 495.

E3 : Oui, y a encore une étape.

P : Donc faut faire une étape de plus, d'accord. Parce que est-ce qu'on obtient toujours 594 ?

E3 : Je sais pas. . .

P : Donc, je vais finir l'étape. Donc finalement, il y a deux questions là, est-ce qu'on obtient toujours 693 et est-ce qu'on obtient toujours 594. Oui, non, pourquoi ? Donc tu le donnes ton contre-exemple ?

E4 : Euh, ben moi j'étais partie avec 503.

P : Avec 503. D'accord.

E4 : Ça fait 530 moins 35.

P : (écrivant) Donc on a 530 moins 35.

E4 : Et ça donne directement 495.

P : OK. Est-ce que tout le monde est d'accord, est-ce que tout le monde est d'accord avec ce calcul-là ?

Murmures : Non, ben oui, quoique, ouais si ça marche...

P : Oui, d'accord ? Vous êtes tous d'accord avec ça ?

Murmures : Si, si, si.

P : Ça veut dire que vous êtes tous d'accord que quand on range les nombres, les chiffres du nombre 503,

[...]

E3 : Enfin, ce que je voulais dire avec ma conjecture, c'est que si jamais il y a deux étapes avant d'arriver à 495, l'étape juste avant ce sera 693.

P : D'accord.

E3 : Mais si y a que une étape, on arrive à 495. Donc ce sera initial, ensuite 495.

P : D'accord.

E3 : Mais si y a deux étapes, ce sera...

P : En fait, l'étape juste avant, c'est pas vraiment, s'il-te-plaît, l'étape juste avant, c'est pas vraiment 693, il y a 594, OK ?

E3 : Oui, ensuite 594. J'ai même trouvé après trois exemples avec 792, ou 297, pareil. Enfin des nombres formés de 9, 7 et 2.

P : D'accord. Donc ce que tu as remarqué, c'est que, en fait, si on sait combien on a d'étapes avant, on va pouvoir déterminer tous les...

E3 : Ouais.

P : Tous les résultats qu'on obtient avant.

E3 : Oui.

P : Ok, donc ça vous pouvez le tester, est-ce-que, donc, ce sera quoi la suite que tu obtiens ?

E3 : Euh...je suis pas allé aussi loin mais bon...

P : Donc à la fin on obtient toujours 495, d'accord, mais si on a eu une étape avant, on a obtenu quoi ?

E3 : 594

P : On a obtenu 594. Si on a une étape avant, on a obtenu quoi ?

E3 : 693

P : 693, et si on a une étape avant, on a obtenu quoi ?

E3 : 792, ou 297

P : Alors 792 ou...

E3 : 792, moi j'ai 792.

P : 792. De toutes façons c'est une conjecture d'accord. On s'intéressera à la démonstration plus tard. Oui ?

E6 : Euh, c'est juste pour, pour cette conjecture-là j'ai un résultat où j'ai 396, et pas 693.

E3 : (inaudible)

P : Donc ta conjecture, c'est que si on a une étape avant, on obtient 792, 693, 594, et 495, d'accord. Donc ça c'est vrai si on a cinq étapes. Si on a quatre étapes, on a une première étape où on sait pas ce qui se passe puis t'obtiens ces trois nombres-là (désigne "693, 594 et 495" au tableau) ; si on a trois étapes, on sait pas ce qui se passe à la première puis ces deux-là (désigne "594 et 495" au tableau), et si on a que deux étapes, on sait pas ce qui se passe à la première, mais à la deuxième on obtient ça (désigne 495).

E3 : C'est ça !

P : Ok. Donc effectivement c'est une conjecture. Donc ça on s'intéressera éventuellement à la preuve de cette conjecture la séance prochaine. Euh, ok. Donc est-ce que vous avez d'autres remarques ? Que ce soit sur les conjectures

à établir à la première question ou sur les tests d'arrêts de la seconde question. Oui ?

E6 : La somme de tous les chiffres de 792, 693, 594, 495, ça fait toujours 18.

P : D'accord. Donc effectivement, la somme de tous les chiffres qu'on obtient dans cette suite-là que tu as conjecturée, ça fait toujours 18. Donc on va peut-être l'écrire : (écrivant) la somme des chiffres, la somme des chiffres, c'est toujours 18. OK. D'autres remarques ? Oui ?

E7 : Et le chiffre du milieu, c'est toujours un 9.

P : Et le chiffre du milieu, c'est toujours 9 ! Oui ? Oui ?

E6 : La soustraction de celui du milieu par celui de gauche, c'est toujours celui de droite.

E8 : Ouais, la somme des extrêmes donne toujours celui du milieu.

E6 : La soustraction du 9, 9 moins 7 c'est 2...

P : D'accord, 9 moins 7 c'est toujours 2, enfin c'est...

E6 : Puis 9 moins 6 c'est 3...

P : D'accord, ça en fait c'est déjà lié à ces deux, à ces deux conjectures-là.

E6 : Ouais, peut-être.

P : Parce que si la somme c'est 18 et que celui du milieu c'est toujours 9...

E6 : Ouais OK.

P : ...la somme des deux autres, ça va toujours être 9 aussi. Ok ? Donc d'autres choses ? Non ? Quelqu'un à un autre test d'arrêt à proposer ou... Non ? C'est bon ?

E8 : Si... On peut faire la somme de tous les chiffres du coup.

P : La somme de tous les chiffres... ?

E8 : Pour... Si le nombre obtenu... Si la somme du chiffre qui...

P : La somme des chiffres du nombre obtenu.

E8 : Si la somme des chiffres du nombre obtenu est égal à 18, un truc comme ça...

P : Alors, ça, là c'est tous les nombres qu'on obtient dans la suite, quand on calcule plusieurs fois l'algorithme, d'accord ? Et on veut pas s'arrêter au premier, la première fois. Par exemple, ici, même même quand on prend la première étape là, la somme de ses chiffres, là, ça va être 18. C'est pas là qu'on veut s'arrêter.

E8 : Ah, on veut s'arrêter à la fin dans tous les cas.

P : On veut s'arrêter quand on obtient un nombre déjà obtenu, d'accord ?

B.1.2 Tests d'arrêt

E1 : Ensuite pour l'étape 5 pour faire un test, on a pensé à faire la différence à chaque fois entre le nombre qu'on a trouvé à la dernière étape et à l'avant-dernière.

P : D'accord.

E1 : Si ça fait 0 on arrête, et si ça fait pas 0, on continue.

P : Pourquoi si ça fait 0 ?

E1 : Parce qu'on doit obtenir deux fois de suite le même nombre. Donc si jamais le dernier moins le précédent ça fait 0, c'est qu'on a obtenu deux fois de suite le même.

P : D'accord.

[...]

P : Et pour la question 2, tu as autre chose à me proposer ?

E2 : Pour la question 2... Ben on a mis un SI...oALORS, dans l'algorithme.

P : Oui.

E2 : Donc on met un SI A-B=495 ou 0, alors l'algorithme...

P : C'est quoi A et B quand tu fais A-B ?

E2 : Si euh, les, euh, le chiffre, euh le chiffre arrangé dans l'ordre croissant et le chiffre arrangé dans l'ordre décroissant, euh, la différence entre ces deux chiffres fait 495 ou 0...

P : D'accord.

E2 : On arrête là.

P : D'accord, donc en fait si le nombre qu'on obtient à l'étape 4.

E2 : Oui c'est ça.

P : C'est là qu'on calcule la différence, vaut 0 on s'arrête.

E2 : 0 ou 495.

P : Euh, 0 ou 495. OK, donc ça c'est une autre possibilité, euh (écrivant au tableau) si le nombre obtenu en 4, euh vaut 0, vaut 0 ou 495, alors on s'arrête.

B.1.3 Choix des variables

E1 : Et ensuite à la fin, on a proposé d'appeler le nombre de départ petit a...

P : Humhum.

E1 : Ensuite les trois chiffres petit a, petit b, petit c. Enfin du coup le premier ce serait grand A.

P : Oui.

E1 : Euh, et puis la différence par exemple h.

[...]

P : Donc après on va rapidement discuter des variables. Donc qui veut me proposer les variables qu'ils ont mis au point ? Oui ?

E3 : Alors, on a A comme nombre initial choisi.

P : Donc, je vais l'écrire là, donc, c'est pour la question 3. Euh, A pour le nombre initial choisi...

E3 : B c'est A arrangé par ordre croissant, enfin le nombre A arrangé par ordre croissant.

P : D'accord. (écrivant) B c'est le nombre, euh, j'écris ce que tu me dis, arrangé par ordre croissant.

E3 : C'est pas initial, c'est A en fait.

P : D'accord.

E3 : Parce que en fait, une fois qu'on aura fait la soustraction du nombre arrangé par ordre décroissant et croissant, on va le remettre dans A. C'est ça que j'avais pensé. Donc du coup, euh, on va prendre A en nombre initial, on va prendre B arrangé en ordre décroissant, C en ordre croissant, on va faire la différence de, euh, C moins B.

P : Oui.

E3 : On va le remettre dans A, et on va refaire avec B et C.

P : D'accord, d'accord. Mais, euh, je vais écrire comme ça d'accord ? Et après, ce que tu en fais de tes variables, ça dépend de l'algorithme que tu vas écrire, d'accord. Pour l'instant, c'est ce que tu vas mettre au départ dans tes variables. Donc tu veux dire que dans A tu vas mettre d'abord le nombre initial choisi, puis après tu mettras la différence, d'accord. Donc ça va être (écrivant) nombre initial choisi et différence. Et donc A c'est le nombre rangé par ordre croissant, non ça c'est B, ok ? et C nombre rangé par ordre décroissant. Ok ? C'est tout ?

E3 : Oui.

P : OK. Est-ce que quelqu'un a autre chose à proposer ?

E9 : Ben si A c'est une variable où c'est un nombre, pour qu'il puisse le ranger dans l'ordre croissant et décroissant, faut d'abord séparer les chiffres des dizaines, des unités et des centaines. Donc il faut trois variables, un pour le chiffre des unités, un pour le chiffre des dizaines et un pour le chiffre des centaines.

[...]

P : Donc (à E3) pour revenir à ce que tu disais, est-ce que tu es d'accord que pour ranger ton nombre dans l'ordre croissant là, les chiffres de ton nombre dans l'ordre croissant...

E3 : Ouais, il faut d'autres variables.

P : Il va falloir des variables pour mettre, dans lesquelles on va mettre les chiffres. Ok ? Donc qu'est-ce que, comment on va les appeler ces variables ?

E9 : Ben j'avais appelé c pour celle des centaines, mais C est déjà utilisé.

P : On peut l'appeler petit c éventuellement.

E9 : On peut l'appeler petit c, petit d et petit u.

P : D'accord. (écrivant) petit c, pour le chiffre des centaines. Le chiffre des centaines de A, c'est ça ?

E9 : Euh, oui.

P : Le chiffre des centaines de A. Petit d, le chiffre des dizaines de grand A. Et petit u pour le chiffre des unités, c'est ça ?

E9 : Oui.

B.1.4 Un traitement sur les chiffres ?

E3 : Dans ce cas ce serait peut-être plus utile de voir dans l'entrée les chiffres, lui dire, on rentre le chiffre des centaines, on rentre le chiffre des dizaines, on rentre le chiffre des unités. Ça empêcherait de faire une décomposition à la machine.

P : D'accord. Donc effectivement, est-ce que tout le monde a compris ce qu'il propose là ? Notre algorithme, il nous dit : première, premier, premier point, c'est choisir un nombre entier de trois chiffres. Et ce que nous propose...

E3 : Thibault

P : Ce que nous propose Thibault, c'est plutôt que de rentrer un nombre entier avec trois chiffres, de rentrer directement les trois chiffres.

E3 : Humhum.

P : D'accord ? Donc plutôt que de rentrer 754, rentrer 7, 5, 4. D'accord. Donc du coup on aurait plus cette variable-là (désigne A), c'est ça ?

E3 : Oui.

P : Et on rentrerait directement ces variables-là (désigne c, d, u). Donc effectivement, c'est une solution.

E8 : Mais après pour le traitement de la soustraction, ça va être difficile. Puisque...

P : Alors, pour le traitement de la soustraction, ça va être difficile.

E8 : De la différence.

P : On va voir.

O : On dit au départ, choisir un nombre entier de trois chiffres.

P : Oui, voilà.

O : Tu dis pas, je sais bien qu'on dit 9,3 au lieu de dire 93, mais, voilà.

P : D'accord.

O : C'est plus, on respecte plus tout à fait le contrat qui...

P : D'accord. Donc, euh, mais ce serait, ce serait une solution pour traiter l'algorithme, et c'est vrai, (à E3) tu as raison, ça poserait pas les mêmes problèmes. Ça en poserait éventuellement d'autres comme par exemple ce qu'a dit ton camarade (E8), ça serait de traiter la soustraction. Parce que si on a pas..., quand on a des nombres, l'algorithme il sait bien faire la soustraction de nombres, d'accord ? Mais si on a uniquement les chiffres, c'est plus compliqué. D'accord ? Oui ?

E11 : On peut choisir le chiffre c fois 100 plus d fois 10 plus u.

E8 : Oui, oui, c'est faisable.

P : Alors, on peut aussi reconstituer le nombre à partir des chiffres, ce serait une solution. Mais bon, l'algorithme...

O : Non, mais..

P : Moi, je suis plutôt contente qu'il dise ça.

O : Oui, je suis d'accord avec toi, mais je veux dire, ça respecte pas exactement...

P : Voilà. L'algorithme, l'entrée de l'algorithme, c'est un nombre à trois chiffres, donc on va garder un nombre à trois chiffres, mais c'est une idée intéressante qui effectivement change les problèmes qu'on aura à traiter. D'accord ? Mais pour l'instant on va garder un nombre à trois chiffres

B.1.5 Identification des étapes

P : Donc, la, la suite, le but de ce qu'on va faire à la suite, c'est de programmer cet algorithme d'accord. Et on va essayer après de programmer sur AlgoBox. Donc, pour pouvoir le programmer, euh, je vais, je vais vous demander, d'après vous quelles sont, quelles vont être les étapes et les petits morceaux de programmes qu'on va avoir besoin de savoir faire, de faire faire à l'ordinateur pour pouvoir programmer cet algorithme. Oui ?

E3 : La décomposition du nombre initial choisi en chiffre des centaines, chiffre des dizaines, chiffre des unités.

P : Humhum.

E3 : Ensuite l'arranger en ordre décroissant, l'arranger en ordre croissant. Faire la soustraction. Et ensuite (inaudible)

P : D'accord. Alors, est-ce que vous êtes tous d'accord avec ça ? Oui ?

E2 : Parce que, on va obtenir justement une variable avec trois chiffres, donc une variable qui va nous donner notre nombre, puis il faut qu'on la... il faut qu'on la redivise en trois variables pour pouvoir la ranger en ordre croissant, décroissant.

P : Oui, donc je pense que c'est ce qu'a proposé ton camarade, quand tu as dit, la première chose à faire, tu as dit que c'était quoi ?

E3 : Décomposer les chiffres.

P : D'accord ? Ça ça va être la première étape. Oui ? Donc effectivement, ce que dit Thibault, c'est que la première... Donc la première chose qu'on va avoir à faire, ça va être de décomposer le nombre qu'on nous donne en chiffres. Est-ce que vous êtes tous d'accord avec ça ? Pourquoi est-ce qu'on va avoir besoin de faire ça ? Parce que vous finalement, quand vous faites, est-ce que vous vous avez décomposé en chiffres ? Oui ?

E3 : L'ordinateur pour arranger les chiffres en ordre décroissant, ça va être important.

P : D'accord. Mais est-ce que vous vous l'avez fait là ? Décomposer en chiffres ?

Murmures

P : Voilà, vous l'avez fait dans votre tête en fait. D'accord ? Parce que vous vous savez passer de 754 à 457 sans vous poser de questions là-dessus. D'accord ? Vous êtes tous d'accord avec ça ? Par contre, l'ordinateur, lui, ben, il faut qu'il sache exactement quel type de tâche il doit effectuer. D'accord ? Donc lui, il peut pas le faire de tête en quelque sorte. Ok ? Donc, est-ce que vous avez des questions là-dessus ? Est-ce que vous êtes tous d'accord que l'ordinateur, lui, il va avoir besoin de passer aux chiffres ? Oui, Ok. Donc, je vais écrire les étapes qu'on va avoir besoin de faire au tableau.

[. . .]

P : Donc je vous distribue la suite de ce qu'on va faire aujourd'hui. Donc la première question, c'est celle que je vous ai posée d'accord. C'est quelles sont à effectuer pour programmer cet algorithme. Donc, on va, on va mettre ça au point ensemble à partir de ce qu'a dit Thibault. Donc les étapes à faire, on est tous d'accord que la première à faire, ça va être de, de décomposer le nombre initial choisi, le fameux nombre A en chiffres. Donc étape 1 (écrit au tableau) décomposer le nombre choisi en chiffres. D'accord ? Donc qu'est-ce qu'on va avoir comme étape après ? Une fois qu'on a les chiffres, qu'est-ce qu'on va faire de ces chiffres ? Oui ?

E13 : Il va falloir faire des tests les uns par rapport aux autres, pour voir lequel est le plus petit, lequel est le plus grand. Les comparer. . .

P : Donc des tests par rapport aux autres pour savoir lequel est le plus petit, lequel est le plus grand. De manière générale, qu'est-ce qu'on va vouloir faire sur ces chiffres ? Oui ?

E1 : Les classer.

P : Oui donc ?

E1 : Les classer dans l'ordre.

P : Les classer dans l'ordre ! Ok ? Vous êtes d'accord ? C'est ça qu'il va falloir faire. Les classer dans l'ordre croissant ou décroissant, mais en tout cas les classer. Ok ? Donc finalement, ça va être quoi ? Est-ce que vous savez comment on appelle ça ? Quand on classe des choses comme ça, ce qu'on va vouloir faire, en fait, ça va être les trier. D'accord ? Donc c'est ce qu'on va vouloir faire dans une seconde partie, euh, la deuxième étape, ça va être trier les chiffres. D'accord ? Les classer. C'est la même chose. D'accord ? Là on s'occupe plus du nombre qu'on a choisi au départ, ok, on s'occupe uniquement des chiffres qu'on a obtenus à l'étape 1. Ok, donc là on trie les chiffres, donc cet algorithme-là, il va nous renvoyer les chiffres triés dans le bon ordre, d'accord, enfin dans l'ordre croissant et dans l'ordre décroissant. Et qu'est-ce qu'il va nous rester à faire après ?

E13 : La différence.

P : La différence de quoi ?

E11 : Reconstituer des nombres.

P : Alors, avant de faire la différence, effectivement, il va falloir reconstituer les nombres. Oui ?

E10 : Euh, oui, pour la différence, des nombres triés en ordre décroissant par celui trié en ordre croissant.

P : Voilà. Donc, et ce sur quoi j'insiste, c'est que cet algorithme-là, cette partie-là (désigne étape 2), elle va nous renvoyer des chiffres. D'accord ? Oui ?

E11 : Ben on fait fois 100, fois 10. . .

P : Voilà, donc le but, comme tu le disais, ça va être de reconstruire les nombres. D'accord. Donc finalement, notre troisième algorithme, il va faire quoi ? À partir de trois chiffres, il va reconstituer le nombre associé. D'accord. Donc (écrivant) étape 3 : Passer des chiffres au nombre. Ok ? Donc en fait ces trois étapes-là, elles vont pouvoir être traitées indépendamment les unes des autres. Vous êtes d'accord avec ça ? Et c'est en les mettant les unes à la suite des autres qu'on va pouvoir, euh, faire notre algorithme. Ok ? Donc notre objectif aujourd'hui, ça va être d'écrire des algorithmes qui correspondent à chacune de ces étapes et de les programmer sur AlgoBox. D'accord ? Donc vous êtes par groupe, chaque groupe va avoir une étape qui va lui être attribuée, d'accord, et va devoir écrire l'algorithme qui correspond et le programmer. Ok ? Donc pour que tout le monde soit, pour que ce soit bien clair sur ce que font les algorithmes, je vais donner des exemples. Donc par exemple, cet algorithme-là (désigne étape 1), qu'est-ce que je veux qu'il fasse ? Je veux que si on entre euh, 538, par exemple, d'accord, on renvoie 5, 3, 8. Vous êtes bien d'accord ? C'est ça qu'on veut. On passe du nombre aux chiffres. Ok. Cet algorithme-là (désigne étape 2), je veux qu'il les trie. Donc je vais avoir besoin de deux choses, je vais avoir besoin des chiffres dans l'ordre croissant et des chiffres dans l'ordre décroissant. Donc si on rentre 5, 3, 8, on renvoie quoi ? Donc c'est des virgules qui séparent les chiffres, hein, c'est pas... On renvoie 3, 5, 8 et 8, 5, 3. D'accord ?

E8 : On renvoie les deux, les deux...

P : Oui !

E8 : Les deux tris d'un coup.

P : Oui.

E8 : D'accord.

P : D'accord ? Vous essayez de faire un algorithme qui fasse les deux d'un coup. Ok ? Et la dernière étape, je veux qu'il fasse quoi l'algorithme ? Je veux que si on entre, du coup 3, 5, 8, on renvoie 358. Ok ? Est-ce que vous avez des questions sur euh, sur ce que font ces algorithmes ? Oui ?

E8 : Au niveau des variables, on se charge de créer autant de variables qu'on veut ?

P : Alors, voilà. Alors au niveau des variables, vous allez chacun avoir à programmer un algorithme, d'accord, une étape, qui va être indépendante. Donc une étape sans regarder ce qui va se passer pour les autres étapes ou dans le reste de l'algorithme. Et c'est à vous de choisir les variables dont vous aurez besoin pour faire cette étape-là. Oui ?

B.1.6 Question au sujet de l'étape 1

E14 : J'ai une question pour l'étape 1, est-ce que il doit afficher les chiffres dans l'ordre du, de l'entrée ? Ou il peut mettre 8, 3, 5 ?

P : Euh, alors, c'est une question intéressante. En fait, euh, est-ce qu'on en a besoin que ce soit dans l'ordre ? Pour la suite de l'algorithme ?

Classe : Non, non.

P : Finalement non, puisqu'on va les reclasser après, on va les retrier après. Donc on en aurait pas besoin. D'accord ? Euh, donc...

O : C'est pas vraiment demandé dans cette étape.

P : D'accord, donc il renvoie pas forcément dans l'ordre. Là, l'ordre n'importe pas. D'accord. Je veux juste qu'il y ait les trois chiffres.

O : Il a extrait les chiffres.

P : Voilà, il a extrait les chiffres, l'ordre n'importe pas. D'accord, c'est un ensemble non ordonné. Par contre là (désigne étape 3), évidemment oui !

O : Oui parce que sinon on rentre déjà dans l'algorithme de tri.

P : Oui, oui. Mais ce qu'elle veut dire c'est...

O : Non, mais j'ai bien compris.

P : Vous le renvoyez comme vous voulez, y a pas de soucis. Par contre là (désigne étape 3), par contre là c'est important ! De là à là, l'ordre importe. Ok ? Là, non, puisqu'on en aura pas besoin, par contre dans cette étape-là, oui. Ok ?

B.1.7 Des question au sujet d'AlgoBox

Ces question ont été posée à l'enseignant ou aux autres élèves du groupe pendant la phase de programmation des étapes.

E1 : On peut faire une troncature sur AlgoBox ?

[...]

E2 : Parce que je sais que ça (%) sur AlgoBox, quand tu fais ça, par exemple si tu fais n% d, c'est le reste de la division euclidienne de n par d.

[...]

E3 : Ensuite faut voir ce que AlgoBox est capable de faire.

E2 : Je sais pas comment on fait le quotient sur AlgoBox.

[...]

E2 : Parce que je sais que j'ai fait un programme avec quotient et reste sur la calculette et tu fais INT, partie entière.

[...]

E2 : Après y a forcément un truc INT sur AlgoBox, pour prendre la partie entière, je vois pas pourquoi y aurait pas.

[...]

E4 : Est-ce que dans AlgoBox y a une fonction qui repère, euh, le plus petit d'une liste, par exemple ?

B.2 Deuxième séance

B.2.1 Et ou Ou ?

E1 : Donc, tant que A est différent de 95 ou différent de 0,

E2 : 495.

E1 : Euh 495, j'ai dit quoi ?

E2 : T'as dit 95.

P : Oui.

[...]

P : Est-ce que quelqu'un a une autre méthode qui utilise des étapes différentes ou des choses différentes ? Oui ?

E3 : Hum, d'abord c'est lire A.

P : Oui.

E3 : Ensuite, tant que A différent de 0 et A différent de 495, euh, c'est égal à partie entière de A divisé par 100.

P : Oui.

[...]

P : Alors, juste un dernier point, donc, vous les filles devant vous m'avez dit que le test qu'on fait là c'est A différent de 495 ou A différent de 0, au fond, tu m'as dit...

E3 : A différent de 0 et A différent de 495.

P : Alors, qu'est-ce qui est correct, qu'est-ce qui ne l'est pas ? Est-ce que les deux sont corrects ? Est-ce que ça fait la même chose si on met un et ou un ou ici ? Oui ?

E7 : Ben moi ça me parait bizarre qu'un nombre soit égal à deux nombres différents.

P : Voilà, donc, quand tu dis le et, tu vois, un nombre qui est différent de deux nombres, c'est, c'est vrai, tu vas trouver des nombres qui sont différents de 495 et différents de 0. D'accord ? Mais tu vas jamais trouver, tu vas jamais satisfaire la condition être égal à 495 et à 0. Est-ce que tu vois ce que je veux dire ? Finalement, ton algorithme, il s'arrêtera quand tu obtiendras un nombre égal à 495 et à 0, d'accord ? Et ça tu vas pas pouvoir le trouver. Tu es d'accord.

E3 : On l'a testé comme ça, ça marchait. On l'a fait fonctionner et ça a marché.

P : Tu l'as fait tourner et ça marche...

O : Tu l'as fait... Ça devrait pas marcher.

P : Oui, mais, AlgoBox il a des conditions d'évaluation des...

E8 : Sinon ça marche pas si on met A égal à 495 et A égal à 0.

O : T'as fait égal ou t'as mis euh...

E3 : J'ai mis différent.

O : Oui, c'est bien différent.

P : Alors attendez,

O : Ben s'il est différent de 495, il est automatiquement différent de 0.

E8 : Si on prend 496, il est différent de 495 et il est différent de 0.

O : Ouais, non, c'est vrai, ça doit marcher...

P : Oui mais si on a 0, ça marche pas...

O : Si on a 0, ça marche pas.

P : Bon, on va y réfléchir, c'est une question à laquelle il faut réfléchir en tout cas.

O : Moi j'aurais mis ou, mais c'est vrai que...

P : Il me semble, mais, euh, on va y réfléchir. Donc, je vous laisse passer sur les ordinateurs et programmer vos algorithmes, donc vous gardez ce que vous avez mis là, là où moi j'ai mis des pointillés, vous gardez ce que vous avez fait. Que ce soit avec des tests ou avec des listes, avec des max, des min ou des choses comme ça, vous gardez ce que vous avez fait.

Chapitre C

Travaux d'élèves

C.1 Programmation des étapes

C.1.1 Étape 1 : extraction des chiffres

En utilisant la partie entière

```
1   VARIABLES
2     A EST_DU_TYPE NOMBRE
3     c EST_DU_TYPE NOMBRE
4     d EST_DU_TYPE NOMBRE
5     u EST_DU_TYPE NOMBRE
6   DEBUT_ALGORITHME
7     LIRE A
8     c PREND_LA_VALEUR floor(A/100)
9     d PREND_LA_VALEUR floor((A-c*100)/10)
10    u PREND_LA_VALEUR (A-c*100-d*10)
11    AFFICHER c
12    AFFICHER d
13    AFFICHER u
14  FIN_ALGORITHME
```

En utilisant la fonction %

```
1   VARIABLES
2     N EST_DU_TYPE NOMBRE
3     c EST_DU_TYPE NOMBRE
4     d EST_DU_TYPE NOMBRE
5     u EST_DU_TYPE NOMBRE
6   DEBUT_ALGORITHME
7     LIRE N
8     u PREND_LA_VALEUR N%10
9     N PREND_LA_VALEUR (N-u)/10
10    d PREND_LA_VALEUR N%10
11    c PREND_LA_VALEUR (N-d)/10
12    AFFICHER c
13    AFFICHER d
```

```
14    AFFICHER u
15  FIN_ALGORITHME
```

En utilisant la partie entière et le modulo

```
1   VARIABLES
2     A EST_DU_TYPE NOMBRE
3     u EST_DU_TYPE NOMBRE
4     d EST_DU_TYPE NOMBRE
5     c EST_DU_TYPE NOMBRE
6     j EST_DU_TYPE NOMBRE
7   DEBUT_ALGORITHME
8     LIRE A
9     c PREND_LA_VALEUR floor(A/100)
10    j PREND_LA_VALEUR A%100
11    d PREND_LA_VALEUR floor(j/10)
12    u PREND_LA_VALEUR j%10
13    AFFICHER "Le chiffre des centaines est"
14    AFFICHER c
15    AFFICHER "Le chiffre des dizaines est"
16    AFFICHER d
17    AFFICHER "Le chiffre des unites est"
18    AFFICHER u
19  FIN_ALGORITHME
```

C.1.2 Étape 2 : tri des chiffres

En faisant 6 tests successifs

```
1   VARIABLES
2     a EST_DU_TYPE NOMBRE
3     b EST_DU_TYPE NOMBRE
4     c EST_DU_TYPE NOMBRE
5     g EST_DU_TYPE NOMBRE
6     h EST_DU_TYPE NOMBRE
7     i EST_DU_TYPE NOMBRE
8   DEBUT_ALGORITHME
9     LIRE a
10    LIRE b
11    LIRE c
12    SI (a<=b<=c) ALORS
13      DEBUT_SI
14      g PREND_LA_VALEUR a
15      h PREND_LA_VALEUR b
16      i PREND_LA_VALEUR c
17      FIN_SI
18    SI (a<=c<=b) ALORS
19      DEBUT_SI
20      g PREND_LA_VALEUR a
21      i PREND_LA_VALEUR b
22      h PREND_LA_VALEUR c
```

```
23      FIN_SI
24    SI (b<=a<=c) ALORS
25      DEBUT_SI
26      h PREND_LA_VALEUR a
27      g PREND_LA_VALEUR b
28      i PREND_LA_VALEUR c
29      FIN_SI
30    SI (b<=c<=a) ALORS
31      DEBUT_SI
32      g PREND_LA_VALEUR b
33      h PREND_LA_VALEUR c
34      i PREND_LA_VALEUR a
35      FIN_SI
36    SI (c<=a<=b) ALORS
37      DEBUT_SI
38      g PREND_LA_VALEUR b
39      h PREND_LA_VALEUR c
40      i PREND_LA_VALEUR a
41      FIN_SI
42    SI (c<=b<=a) ALORS
43      DEBUT_SI
44      g PREND_LA_VALEUR c
45      h PREND_LA_VALEUR b
46      i PREND_LA_VALEUR a
47      FIN_SI
48    AFFICHER g
49    AFFICHER h
50    AFFICHER i
51    AFFICHER i
52    AFFICHER h
53    AFFICHER g
54  FIN_ALGORITHME
```

En faisant des test imbriqués

```
1   VARIABLES
2     a EST_DU_TYPE NOMBRE
3     b EST_DU_TYPE NOMBRE
4     c EST_DU_TYPE NOMBRE
5     L1 EST_DU_TYPE LISTE
6     L2 EST_DU_TYPE LISTE
7   DEBUT_ALGORITHME
8     LIRE a
9     LIRE b
10    LIRE c
11    SI (a>=b) ALORS
12      DEBUT_SI
13      SI (a>=c) ALORS
14        DEBUT_SI
15        L1[0] PREND_LA_VALEUR a
```

```
16        SI (b>=c) ALORS
17          DEBUT_SI
18          L1[1] PREND_LA_VALEUR b
19          L1[2] PREND_LA_VALEUR c
20          FIN_SI
21          SINON
22            DEBUT_SINON
23            L1[1] PREND_LA_VALEUR c
24            L1[2] PREND_LA_VALEUR b
25            FIN_SINON
26        FIN_SI
27        SINON
28          DEBUT_SINON
29          L1[0] PREND_LA_VALEUR c
30          L1[1] PREND_LA_VALEUR a
31          L1[2] PREND_LA_VALEUR b
32          FIN_SINON
33      FIN_SI
34      SINON
35        DEBUT_SINON
36        SI (b>=c) ALORS
37          DEBUT_SI
38          L1[0] PREND_LA_VALEUR b
39          SI (a>=c) ALORS
40            DEBUT_SI
41            L1[1] PREND_LA_VALEUR a
42            L1[2] PREND_LA_VALEUR c
43            FIN_SI
44            SINON
45              DEBUT_SINON
46              L1[1] PREND_LA_VALEUR c
47              L1[2] PREND_LA_VALEUR a
48              FIN_SINON
49          FIN_SI
50          SINON
51            DEBUT_SINON
52            L1[0] PREND_LA_VALEUR c
53            L1[1] PREND_LA_VALEUR b
54            L1[2] PREND_LA_VALEUR a
55            FIN_SINON
56        FIN_SINON
57    AFFICHER "Ordre decroissant :"
58    AFFICHER L1[0]
59    AFFICHER L1[1]
60    AFFICHER L1[2]
61    L2[0] PREND_LA_VALEUR L1[2]
62    L2[1] PREND_LA_VALEUR L1[1]
63    L2[2] PREND_LA_VALEUR L1[0]
64    AFFICHER "Ordre croissant :"
65    AFFICHER L2[0]
```

```
66    AFFICHER L2[1]
67    AFFICHER L2[2]
68  FIN_ALGORITHME
```

En utilisant la structure de liste et les fonctions min et max

```
1   VARIABLES
2     C EST_DU_TYPE NOMBRE
3     D EST_DU_TYPE NOMBRE
4     U EST_DU_TYPE NOMBRE
5     A EST_DU_TYPE LISTE
6     B EST_DU_TYPE LISTE
7     I EST_DU_TYPE NOMBRE
8   DEBUT_ALGORITHME
9     LIRE C
10    LIRE D
11    LIRE U
12    A[0] PREND_LA_VALEUR C
13    A[1] PREND_LA_VALEUR D
14    A[2] PREND_LA_VALEUR U
15    B[2] PREND_LA_VALEUR ALGOBOX_MAXIMUM(A,0,2)
16    B[0] PREND_LA_VALEUR ALGOBOX_MINIMUM(A,0,2)
17    SI (C != B[2] ET C !=B[0]) ALORS
18      DEBUT_SI
19      B[1] PREND_LA_VALEUR C
20      FIN_SI
21      SINON
22        DEBUT_SINON
23        SI (D != B[2] ET D !=B[0]) ALORS
24          DEBUT_SI
25          B[1] PREND_LA_VALEUR D
26          FIN_SI
27          SINON
28            DEBUT_SINON
29            B[1] PREND_LA_VALEUR U
30            FIN_SINON
31        FIN_SINON
```

C.1.3 Étape 3 : reconstitution du nombre

```
1   VARIABLES
2     u EST_DU_TYPE NOMBRE
3     c EST_DU_TYPE NOMBRE
4     d EST_DU_TYPE NOMBRE
5     i EST_DU_TYPE NOMBRE
6     k EST_DU_TYPE NOMBRE
7     S EST_DU_TYPE NOMBRE
8   DEBUT_ALGORITHME
9     LIRE c
10    LIRE d
11    LIRE u
```

```
12    i PREND_LA_VALEUR c*100
13    k PREND_LA_VALEUR d*10
14    S PREND_LA_VALEUR i+k+u
15    AFFICHER "Le nombre obtenu est"
16    AFFICHER S
17  FIN_ALGORITHME
```

Chapitre D

Énoncés et travaux menés par d'autres enseignants

D.1 Extrait du manuel math'x

D.2 Travail mené en seconde par Françoise

ALGORITHME DE KAPREKAR - Expérimentation en classe de seconde (Françoise)

1. **Exercice donné en temps libre**

On donne l'algorithme ci-dessous :

ENTRÉE	Saisir un entier N à 3 chiffres distincts deux à deux.
TRAITEMENT	Répéter 5 fois : G prend la valeur du plus grand nombre que l'on peut écrire avec les 3 chiffres de N P prend la valeur du plus petit nombre que l'on peut écrire avec les 3 chiffres de N D prend la valeur $G - P$ N prend la valeur D
SORTIE	Afficher N

(a) Quelles sont les variables informatiques utilisées dans cet algorithme ?

(b) Faire tourner cet algorithme en indiquant, pour chaque étape, le contenu de chaque variable lorsque l'on saisit comme entrée le nombre 718.

(c) Faire tourner cet algorithme en indiquant, pour chaque étape, le contenu de chaque variable en saisissant comme entrée un nombre entier à 3 chiffres de votre choix.

(d) **Émettre une conjecture.**

2. **Bilan en classe (en demi-groupe)**

Quelques élèves avaient confondu « chiffres »et « nombres ». Quelques élèves avaient omis de « Répéter 5 fois ». Enfin 1 élève avait ajouté la valeur de D à la valeur de N, au lieu de remplacer la valeur de N par celle de D. Néanmoins presque tous les élèves avaient pu conjecturer que : « Quel que soit le nombre N saisi en entrée, le nombre N affiché en sortie est égal à 495. »

Pour démontrer cette conjecture, les élèves ont proposé, dans le premier groupe, d'étudier tous les cas possibles avec un tableur et dans le deuxième groupe, « de prendre des lettres ».

(a) **Étude de tous les cas possibles avec un tableur**

Afin que les élèves puissent ensuite travailler en autonomie, je leur ai d'abord présenté quelques fonctions du tableur :

- ENT : pour prendre la partie entière d'un nombre.
- STXT : pour extraire une chaîne de caractères (ici pour extraire un chiffre d'un nombre).
- CNUM : pour transformer du format texte au format numérique.
- GRANDE.VALEUR : pour renvoyer la kième plus grande valeur d'une série de nombres.

Les élèves ont ensuite travaillé individuellement avec un tableur.

L'extraction des différents chiffres s'est faite pour la plupart avec la fonction « STXT », sinon avec la fonction « ENT » (en commençant par extraire le chiffre des centaines). Le classement par ordre croissant (ou décroissant) s'est fait sans problème. En revanche, l'écriture d'un nombre à partir des trois chiffres présente une réelle difficulté pour les élèves.

(b) **Démonstration « avec des lettres »**

En notant x, y et z les chiffres de N, x étant le plus petit et z le plus grand, on a $G = \overline{zyx}$ et $G = \overline{xyz}$, la difficulté pour les élèves étant de calculer D : cette difficulté est le même que dans la méthode avec un tableur, à savoir l'écriture d'un nombre à partir de ses trois chiffres. Le calcul conduit à $D = 99z - 99x$, puis, en étudiant les différents cas, on obtient différentes valeurs envisageables pour D. Il reste alors encore à traiter ces différents cas.

L'heure s'est terminée sans que les élèves aient finalisé la démonstration, qu'il s'agisse de la démonstration « avec des lettres » ou de la démonstration « avec un tableur ».

3. **Exercice donné en temps libre : « Démontrer la conjecture de la question (d). »**

(a) **Étude de tous les cas possibles avec un tableur**

• Une élève a fourni le fichier intitulé « Kaprekar-eleve-1.ods », accompagné du commentaire ci-dessous :

Tous les nombres qui contiennent trois chiffres identiques comme 111, 222, ..., 999 sont des nombres qui s'annulent car il n'y a pas de plus grand ou de plus petit donc au moment de soustraire le plus grand et le plus petit, le résultat est égal à zéro du premier au dernier D.
Je constate que les nombres qui à la fin de l'algorithme se terminent par 594 sont des nombres qui ne contiennent

que deux chiffres différents et qui sont égaux depuis le début car dès le premier « D » ils sont tous égaux à 99. Pour tous les autres nombres l'algorithme affiche 495 au bout de la $5^{ième}$ répétition.

• Un autre élève a fourni le fichier intitulé « Kaprekar-eleve-2.ods », accompagné du commentaire ci-dessous :

On a reconstitué l'algorithme sur un ordinateur à l'aide d'un tableur. On a testé des nombres entiers allant de 102 à 987. On a répété deux fois le processus pour chaque nombre. Dans le premier résultat on voit que beaucoup de nombres sont égaux, certains sont déjà à égaux à 495. Dans le deuxième résultat on voit plus de nombres égaux à 495 que dans le premier. Les nombres les plus proches de 495 dans le premier résultat sont devenus 495 dans le deuxième résultat, et tous les autres résultats se sont rapprochés de 495.

(b) **Démonstration « avec des lettres »**

Un extrait de copie d'élève : *Je démontre que, avec l'algorithme de Kaprekar, quel que soit le nombre choisi, le nombre de sortie sera 495.*

Je remplace les chiffres par les lettres a, b et c.

On donne $a > b > c$. Donc le plus grand nombre que l'on peut écrire avec les 3 chiffres de N est abc et le plus petit nombre P que l'on peut écrire est cba.

$D = G - P$
$= abc - cba$
$= 100 \times a + 10 \times b + c - (100 \times c + 10 \times b + a)$
$= 99a - 99c$
$= 99\ (a - c)$

Donc le nombre D est un multiple de 99.

On sait que $(a - c)$ est un nombre entier compris entre 2 et 9 car $a > c$.

L'élève énumère ensuite les différentes valeurs possibles pour D, et pour ces différentes valeurs possibles, fait « *tourner plusieurs fois l'algorithme* » et obtient toujours 495.

4. **Deuxième bilan en classe**

Deux points sont ainsi abordés, quelle que soit la méthode choisie par les élèves (utilisation d'un tableur ou démonstration « avec des lettres ») :

- l'écriture d'un nombre dans le système de numération en base 10 ;
- la possibilité de réduire le nombre de cas initial à étudier à seulement 8 cas.

D.3 Travail mené en première S

Compte rendu TD algorithme de Kaprekar en 1°S

Situation : classe de 1°S de 34 élèves ; travail en demi-classe par groupes de 2 ou 3.

Première séance : questions 1 et 2 du T.D

Objectifs :

- Ecrire un premier algorithme de tri ;
- Utilisation de « si…sinon ».

Déroulement de la séance :

Les élèves testent l'algorithme sur deux exemples chacun et remarquent qu'on finit toujours par obtenir 495. Peut-être pourrait-on le démontrer par l'algorithmique, en testant tous les cas possibles ?

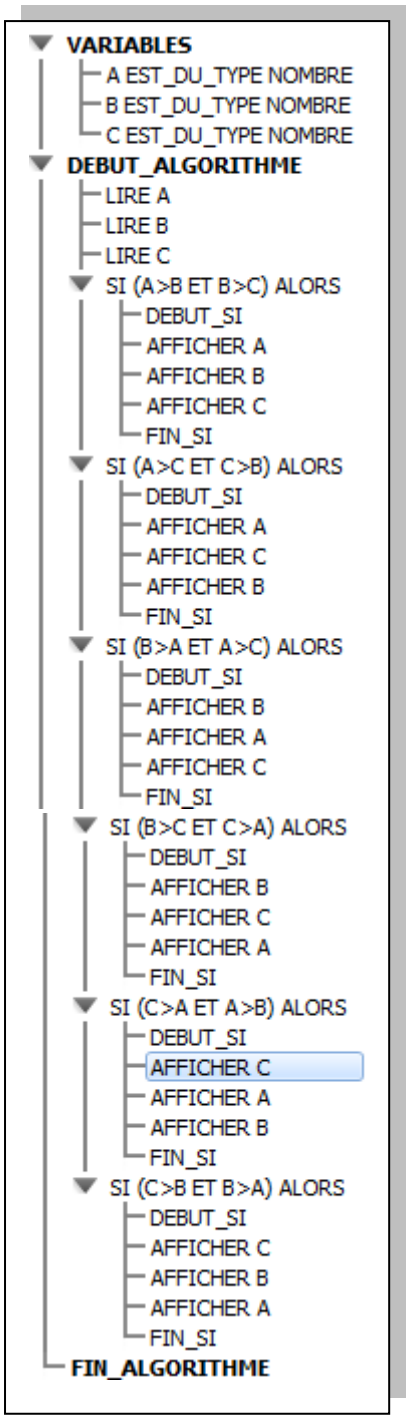
Seuls 4 groupes répondent entièrement à la question2 au cours de la séance et trois autres ont l'algorithme en langage naturel mais pas sous algobox.
Deux groupes pensent répondre à la question en écrivant un algorithme qui ne donne que le plus grand des trois nombres.

Dans l'ensemble, les élèves ne comprennent pas bien l'énoncé : certains essaient de partir d'un nombre de trois chiffres (j'ai donc rajouté le commentaire en italique après la question 1 dans l'énoncé); d'autres ne voient pas du tout ce qu'il faut faire ni comment démarrer. Il est vrai que c'est la première fois que je leur demande de créer un algorithme avec autant de cas à tester sans modèle. Dans les exercices précédents, j'ai souvent donné un algorithme à lire puis à modifier pour répondre petit à petit au problème posé.
D'autre part, la plupart des groupes ne suit pas la consigne : ils ne passent pas par l'étape « écriture sur le papier d'un algorithme en langage naturel » ; ils préfèrent ouvrir tout de suite le logiciel.
Ceux qui ont transposé leur travail sur algobox, confondent souvent « déclarer la variable » et « lire la variable » et même « lire la variable » et « afficher la variable » ce qui révèle sans doute un manque de pratique.

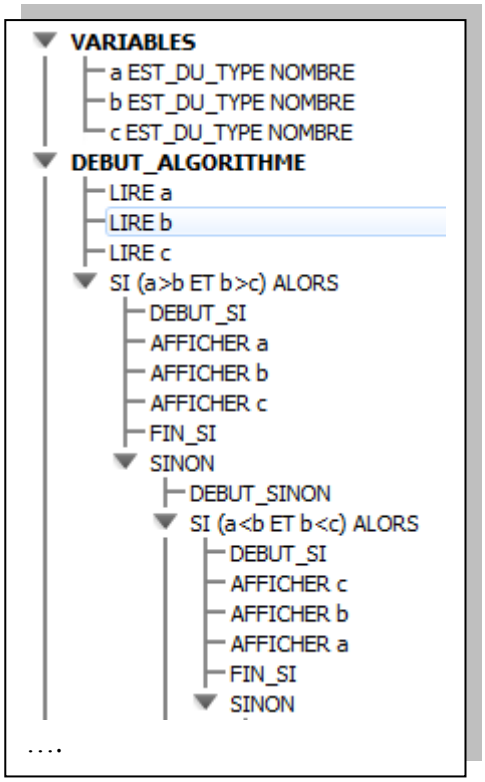
Les algorithmes proposés testent les 6 cas indépendamment les uns des autres ; un seul groupe utilise le « sinon ». En fait, ils l'ont écrit mais ne se sont pas servi de ce que cela leur apportait ; ils ont donc aussi testé les 6 cas les uns après les autres.
Nous commençons alors ensemble un algorithme privilégiant l'utilisation de « sinon » que les élèves termineront seuls. Je leur distribue un corrigé au début de la séance suivante pour qu'ils puissent tous répondre à la question 4.

Travaux d'élèves :

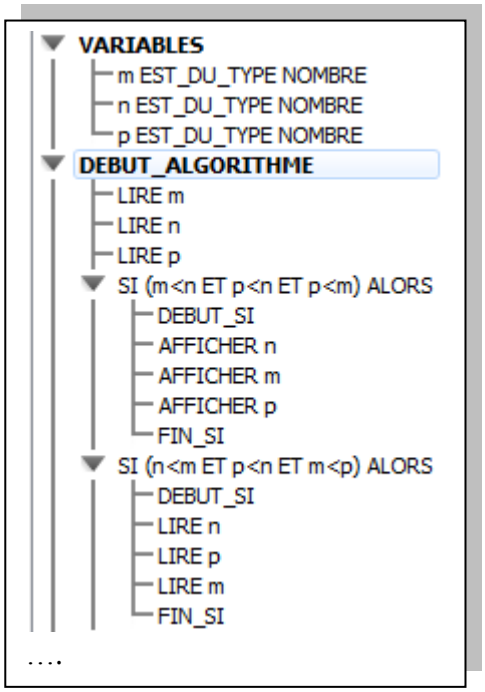
Algorithme de tri décroissant testant les 6 cas indépendamment :



Algorithme (partiel) de tri décroissant dans lequel les élèves pensent utiliser « sinon » :



Confusion entre « lire » et « afficher » :



Deuxième séance : question 3 du T.D

Objectifs :

- Décomposition d'un nombre de trois chiffres ;
- (Re)découvrir la fonction partie entière ou celle donnant le reste dans une division euclidienne.

Déroulement de la séance :

Beaucoup de difficultés pour isoler le chiffre des unités d'un nombre à trois chiffres…Les élèves ne voient pas quelle démarche adopter, quelle opération proposer. Nombreux sont ceux qui pensent qu'il suffit d'entrer les chiffres *a*, *b* et *c* puis de nommer « *abc* » le nombre à trois chiffres pour que le logiciel distingue les trois chiffres.

Deux idées différentes émergent finalement :

- « tant que $x > 10$, x prend la valeur $x - 10$ » ;
- Utiliser la partie entière pour trouver le chiffre des centaines qui est alors $E\left[\frac{n}{100}\right]$.

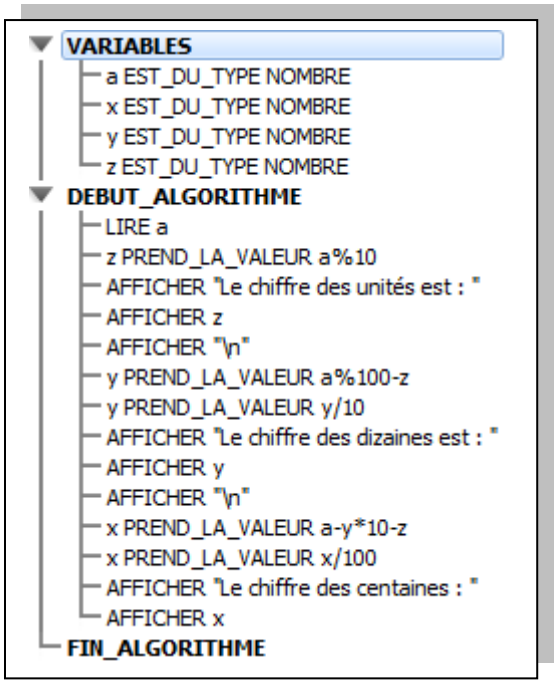
  C'est une fonction que nous avions découverte l'année précédente lors du travail sur l'exercice « enchainement d'entiers »

Je propose d'utiliser une division euclidienne. On découvre ensemble la fonction « *x*%*y* » d'algobox.

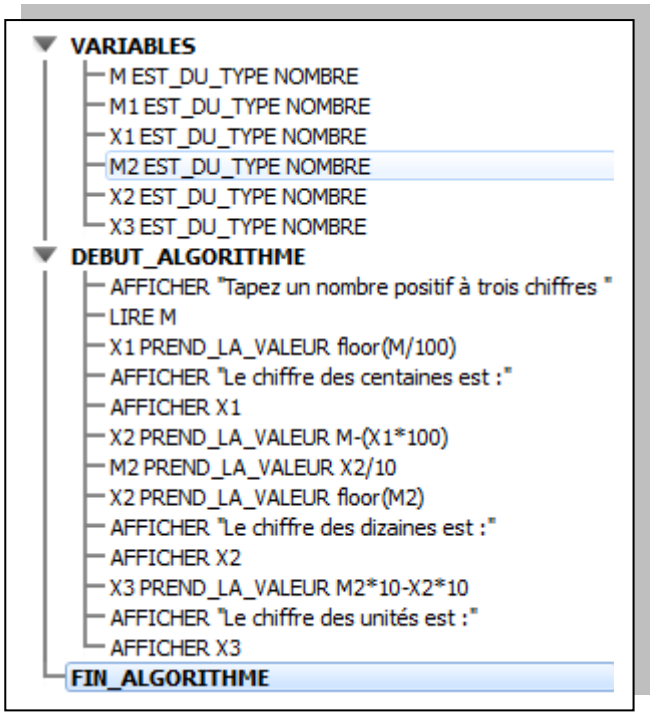
Les élèves choisissent une des trois méthodes citées pour répondre à la question 3. Le travail est à terminer, si besoin, pour la semaine suivante.

Travaux d'élèves :

Algorithme utilisant la division euclidienne :



Algorithme utilisant la partie entière :



Algorithme utilisant des soustractions successives :

```
VARIABLES
    a EST_DU_TYPE NOMBRE
    b EST_DU_TYPE NOMBRE
    c EST_DU_TYPE NOMBRE
    d EST_DU_TYPE NOMBRE
    e EST_DU_TYPE NOMBRE
    f EST_DU_TYPE NOMBRE
    g EST_DU_TYPE NOMBRE
    h EST_DU_TYPE NOMBRE
    i EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
    AFFICHER "Choisir un nombre compris entre 100 et 999"
    LIRE h
    SI (100<=h<=999) ALORS
        DEBUT_SI
        a PREND_LA_VALEUR h
        d PREND_LA_VALEUR h
        g PREND_LA_VALEUR h
        TANT_QUE (a>10) FAIRE
            DEBUT_TANT_QUE
            a PREND_LA_VALEUR a-10
            FIN_TANT_QUE
        AFFICHER "Le chiffre des unités est"
        AFFICHER a
        b PREND_LA_VALEUR a
        TANT_QUE (d>100) FAIRE
            DEBUT_TANT_QUE
            d PREND_LA_VALEUR d-100
            FIN_TANT_QUE
        e PREND_LA_VALEUR d-b
        f PREND_LA_VALEUR e/10
        AFFICHER "Le chiffre des dizaines est"
        AFFICHER f
        g PREND_LA_VALEUR g-d
        i PREND_LA_VALEUR g/100
        AFFICHER "La chiffre des centaines est"
        AFFICHER i
        FIN_SI
        SINON
            DEBUT_SINON
            AFFICHER "Faux. Choisir un nombre compris entre
            FIN_SINON
FIN_ALGORITHME
```

Algorithme mélangeant 2 Méthodes :

```
VARIABLES
    x EST_DU_TYPE NOMBRE
    u EST_DU_TYPE NOMBRE
    d EST_DU_TYPE NOMBRE
    c EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
    LIRE x
    u PREND_LA_VALEUR x
    TANT_QUE (u>10) FAIRE
        DEBUT_TANT_QUE
        u PREND_LA_VALEUR u-10
        FIN_TANT_QUE
    d PREND_LA_VALEUR x
    TANT_QUE (d>100) FAIRE
        DEBUT_TANT_QUE
        d PREND_LA_VALEUR d-100
        FIN_TANT_QUE
    d PREND_LA_VALEUR d/10
    d PREND_LA_VALEUR floor(d)
    c PREND_LA_VALEUR x
    c PREND_LA_VALEUR c/100
    c PREND_LA_VALEUR floor(c)
    AFFICHER "Chiffre des centaines : "
    AFFICHER c
    AFFICHER "Chiffre des dizaines : "
    AFFICHER d
    AFFICHER "Chiffre des unités : "
    AFFICHER u
FIN_ALGORITHME
```

Troisième séance : question 4 du T.D

Objectif :

- Utiliser les questions 2 et 3 pour trouver K(n).

Déroulement de la séance :

Reconstituer le nombre de trois chiffres connaissant ses trois chiffres pose peu de problème.

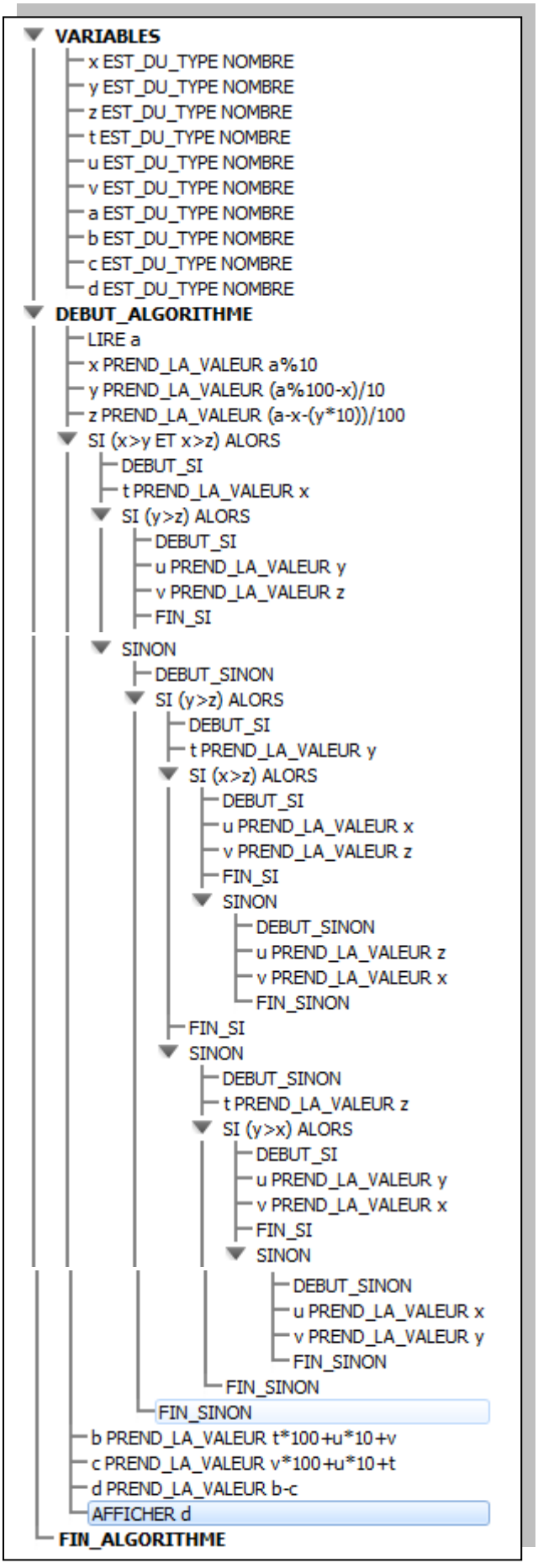
Certains réalisent que, selon les choix de méthodes faits aux questions précédentes, la réponse à la question 4 est plus ou moins rapide ; ils envisagent parfois de modifier leurs algorithmes pour qu'ils soient moins fastidieux à recopier. Ils abordent pour la première fois la question de l'efficacité d'une méthode.

Plusieurs groupes donnent rapidement une réponse à la question 4 ; je leur propose alors de modifier ce dernier algorithme pour montrer que, quelque soit le nombre de 3 chiffres que l'on entre, on obtient 495 par itérations successives.

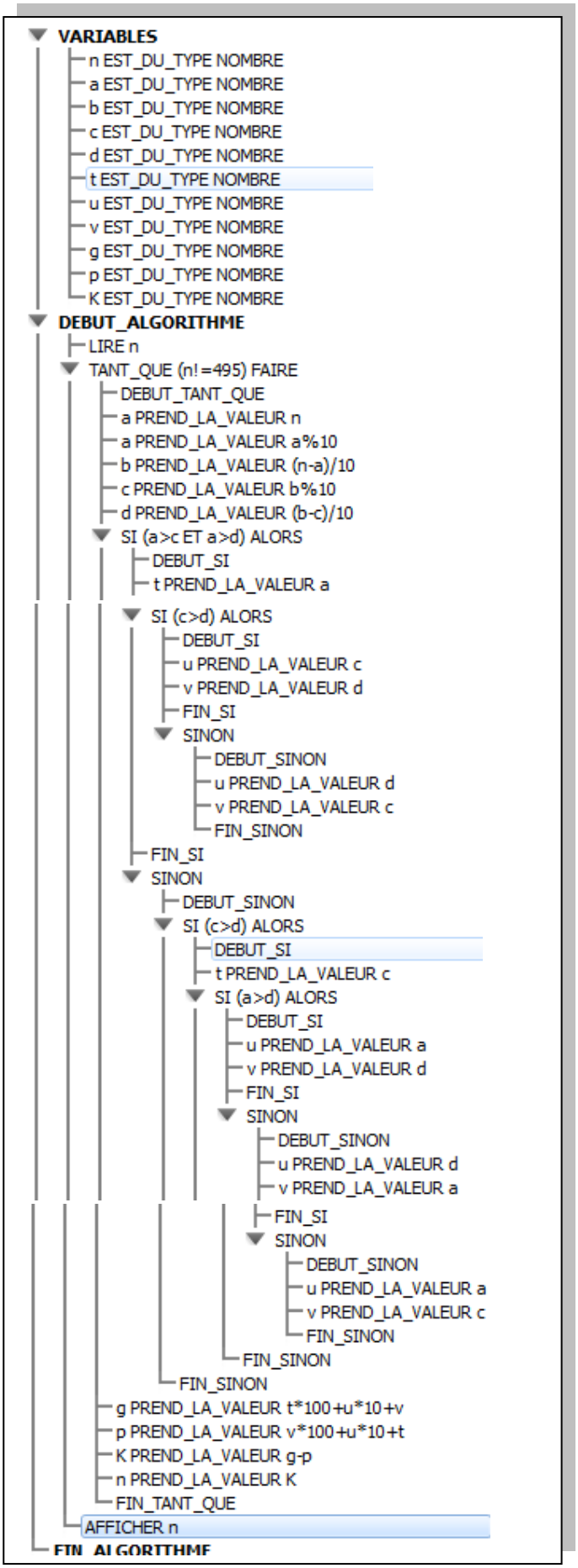
Quelques groupes trouvent une réponse en utilisant comme test d'arrêt « $K(n) \neq 495$ » mais quand je leur fait remarquer qu'il vaudrait mieux trouver un test d'arrêt qui n'utilise pas ce que l'on cherche à montrer, un seul groupe modifie correctement son algorithme.

Travaux d'élèves :

Algorithme répondant à la question 4 :



Algorithme répondant partiellement à la question subsidiaire :



- **VARIABLES**
  - a EST_DU_TYPE NOMBRE
  - x EST_DU_TYPE NOMBRE
  - y EST_DU_TYPE NOMBRE
  - z EST_DU_TYPE NOMBRE
  - g EST_DU_TYPE NOMBRE
  - p EST_DU_TYPE NOMBRE
  - t EST_DU_TYPE NOMBRE
  - u EST_DU_TYPE NOMBRE
  - v EST_DU_TYPE NOMBRE
  - b EST_DU_TYPE NOMBRE
  - c EST_DU_TYPE NOMBRE
- **DEBUT_ALGORITHME**
  - LIRE a
  - b PREND_LA_VALEUR 0
  - c PREND_LA_VALEUR 0
  - z PREND_LA_VALEUR a%10
  - y PREND_LA_VALEUR a%100-z
  - y PREND_LA_VALEUR y/10
  - x PREND_LA_VALEUR a-y*10-z
  - x PREND_LA_VALEUR x/100
  - TANT_QUE (a!=c) FAIRE
    - DEBUT_TANT_QUE
    - z PREND_LA_VALEUR a%10
    - y PREND_LA_VALEUR a%100-z
    - y PREND_LA_VALEUR y/10
    - x PREND_LA_VALEUR a-y*10-z
    - x PREND_LA_VALEUR x/100
    - SI (x>y ET x>z) ALORS
      - DEBUT_SI
      - t PREND_LA_VALEUR x
      - SI (y>z) ALORS
        - DEBUT_SI
        - u PREND_LA_VALEUR y
        - v PREND_LA_VALEUR z
        - FIN_SI
      - SINON
        - DEBUT_SINON
        - u PREND_LA_VALEUR z
        - v PREND_LA_VALEUR y
        - FIN_SINON
      - FIN_SI
    - SINON
      - DEBUT_SINON
      - SI (y>z) ALORS
        - DEBUT_SI
        - t PREND_LA_VALEUR y
        - SI (x>z) ALORS
          - DEBUT_SI
          - u PREND_LA_VALEUR x
          - v PREND_LA_VALEUR z
          - FIN_SI
        - SINON
          - DEBUT_SINON
          - u PREND_LA_VALEUR z
          - v PREND_LA_VALEUR x
          - FIN_SINON
        - FIN_SI
      - SINON
        - DEBUT_SINON
        - t PREND_LA_VALEUR z
        - SI (y>x) ALORS
          - DEBUT_SI
          - u PREND_LA_VALEUR y
          - v PREND_LA_VALEUR x
          - FIN_SI
        - SINON
          - DEBUT_SINON
          - u PREND_LA_VALEUR x
          - v PREND_LA_VALEUR y
          - FIN_SINON
        - FIN_SINON
      - FIN_SINON

...

Algorithme répondant à la question Subsidiaire.

....
- g PREND_LA_VALEUR t*100+u*10+v
- p PREND_LA_VALEUR v*100+u*10+t
- b PREND_LA_VALEUR g-p
- AFFICHER "\n"
- c PREND_LA_VALEUR a
- a PREND_LA_VALEUR b
- AFFICHER b
- FIN_TANT_QUE

FIN_ALGORITHME

D.4 Travail mené en seconde

Compte-rendu d'une expérimentation en classe de seconde

Travail mené avec une classe de seconde sur plusieurs semaines.
Chaque exercice a été donné en devoir en temps libre, (accompagné d'un autre exercice sur un thème différent), dans des devoirs non consécutifs, sur une période s'étendant de novembre à mars.

Exercice N°1. Algorithme de Kaprekar

On s'intéresse ici aux nombres de 3 chiffres, tous distincts. Cet algorithme consiste à associer à un nombre quelconque *n* un autre nombre K(*n*) généré de la façon suivante :
1) On considère les chiffres de *n*. On appelle P le plus petit nombre de 3 chiffres que l'on peut écrire à partir de ces chiffres *(les chiffres de n sont donc alors rangés dans l'ordre croissant)* et G le plus grand nombre que l'on peut écrire dans les mêmes conditions *(les chiffres de n sont donc alors rangés dans l'ordre décroissant)*.
2) On pose K(*n*) = G - P.
3) On itère ensuite le processus avec K(*n*).
Exemple : Si on commence avec 634, on obtient : K(634) = 643 − 346 = 297. En répétant le processus : K(297) = 972 − 279 = 693, puis K(693) = 963 − 369 = 594, K(594) = 954 − 459 = 495, K(495) = 954 − 459 = 495, et on obtient ensuite toujours 495.
Question : effectuer cet algorithme avec 2 entiers différents de 3 chiffres.

Commentaires
Cet énoncé permet juste aux élèves de prendre connaissance de l'algorithme de Kaprekar, et permet aussi d'émettre la conjecture en classe, suite aux différents calculs effectués par les élèves.

Exercice N°2. Étude d'un algorithme

1) Quel est le but de l'algorithme ci-contre ?
2) Écrire un algorithme demandant à l'utilisateur de saisir 3 réels et restituant le plus grand des 3.

Exercice 2, question 1)

```
Variables
    3 variables réelles x, y, z
Traitement
    Saisir x et y.
    Si x < y
        alors z ← y
        sinon z ← x
    Fin Si
Sorties
    Afficher z.
```

Commentaires
La question 1) n'a posé aucun souci particulier.

Plusieurs propositions différentes pour la question 2) :

- **l'algorithme 1.1** teste d'abord si l'un des nombres est supérieur aux deux autres, et si ce n'est pas le cas, il teste quel est le plus grand de ces deux autres. Il comporte un travail éventuel sur la signification du « sinon » dans la boucle : l'exprimer en texte n'est pas compliqué (*x* n'est pas le plus grand entier) mais l'exprimer avec des inégalités nécessite plus de travail (le « et » qui devient « ou »...)
- **l'algorithme 1.2** compare deux entiers et compare ensuite, dans une autre boucle, le plus grand de ces deux-là à l'entier restant. Pas de souci de compréhension mais nécessité d'introduire encore une variable supplémentaire.
- **L'algorithme 1.3** compare deux entiers, puis le plus grand de ces deux-là à l'entier restant, tout ceci à l'aide de « si » imbriqués. Travail sur les « si », « sinon »...

Algorithme 1.1

```
Variables
    4 variables réelles x, y, z, t
Traitement
    Saisir x, y, z.
    Si x > y et x > z
        alors t ← x
        sinon
            si y > z
                alors t ← y
                sinon t ← z
            Fin Si
    Fin Si
Sorties
    Afficher t.
```

Algorithme 1.2

```
Variables
    5 variables réelles x, y, z, t, u
Traitement
    Saisir x, y, z.
    Si x < y
        alors t ← y
        sinon t ← x
    Fin Si
    Si t < z
        alors u ← z
        sinon u ← t
    Fin Si
Sorties
    Afficher u.
```

Algorithme 1.3

```
Variables
    4 variables réelles x, y, z, t
Traitement
    Saisir x, y, z.
    Si x > y
        alors
            si x > z
                alors t ← x
                sinon t ← z
            Fin Si
        sinon
            si y > z
                alors t ← y
                sinon t ← z
            Fin Si
    Fin Si
Sorties
    Afficher t.
```

Exercice N°3. Élaborer un algorithme

Écrire un algorithme permettant de déterminer le chiffre des unités d'un nombre d'un entier quelconque.

Commentaires

Exercice qui a laissé bien des élèves perplexes !
Une proposition de réponse (l'algorithme 3.1) qui a ensuite été amélioré en classe avec l'introduction de la fonction partie entière (notée E ici)

Algorithme 3.1

Variables
1 variable entière x
Traitement
Saisir x.
Tant que $x > 10$
$x \leftarrow x - 10$
Fin Tant que
Sorties
Afficher x.

Algorithme 3.2

Variables
1 variable entière x
Traitement
Saisir x.
$x \leftarrow x - 10 * E\left(\frac{x}{10}\right)$
Sorties
Afficher x.

Exercice N°4. Kaprekar avec un outil informatique

1) Dans le DM..., exercice ..., nous avions effectué un algorithme de calculs sur des nombres entiers à 3 chiffres, appelé algorithme de Kaprekar. Redonner un exemple, et rappeler la conjecture énoncée en classe.
2) Dans le DM... , exercice ..., nous avions vu un algorithme permettant de déterminer le chiffre des unités d'un nombre d'un entier quelconque. Comment obtenir aussi le chiffre des dizaines et le chiffre des centaines d'un nombre entier à 3 chiffres ?
3) Dans le DM..., exercice ..., nous avions vu comment déterminer à l'aide d'un algorithme le plus grand de 3 nombres donnés. Comment classer par ordre décroissant trois nombres donnés ?
4) En déduire un algorithme qui permettrait "d'automatiser" les calculs effectués à la question 1).

Commentaires

Exercice à proposer éventuellement sous une forme moins "directive" suivant la classe concernée pour les questions 2) et 3).
La question 4) est par contre relativement ouverte :

- Combien de calculs effectuer à partir d'un nombre donné ? Test d'arrêt ?
- Possibilité d'effectuer l'algorithme pour tous les nombres entiers et de valider ainsi la conjecture ? Effectuer alors un "tri" des entiers *(3 chiffres tous distincts)* avant l'algorithme ou étudier à posteriori les réponses "non conformes" ?

Cet exercice peut se prolonger par la démonstration à l'aide du calcul littéral, et une comparaison des méthodes utilisées.

Exercice N°5. Kaprekar avec du calcul littéral

On appelle a, b, c les trois chiffres obtenus à partir de l'entier initial dans l'algorithme de Kaprekar, rangés dans l'ordre décroissant.
1) Factoriser l'expression obtenue en effectuant la différence entre le plus grand nombre et le plus petit nombre de 3 chiffres que l'on peut écrire à partir de a, b, c.
2) Quelles valeurs peut prendre $a - c$? Étudier les différents cas. *(raisonnement par disjonction des cas)*
3) Conclure.