# Table des matières

# Introduction

Cette thèse se propose d'apporter des éléments de réponses au problème du fort taux d'échec dans l'enseignement-apprentissage de l'algorithmique à l'ENSET de Libreville au Gabon. L'objectif de ce travail de recherche est d'identifier, comprendre l'origine des difficultés des apprenants dans cet enseignement, et éventuellement, tenter d'y apporter des éléments de solution.

Depuis plusieurs années, à l'École Normale Supérieure de l'Enseignement Technique de Libreville (ENSET), et en dépit des efforts conjugués des enseignants et des apprenants, il est constaté un taux d'échec moyen très élevé dans le module dédié à l'algorithmique et à la programmation. Ce taux d'échec moyen est de l'ordre de quatre-vingt-treize virgule deux pour cent (93,2 %). Ce constat a conduit les responsables pédagogiques à identifier plusieurs écueils susceptibles de justifier ces mauvais résultats. Il s'agit en particulier :

- du temps alloué au module ;
- de la formation initiale des élèves ;
- de l'enseignement-apprentissage de l'algorithmique et de la programmation.

Pour résoudre le problème lié à la durée du module, il a été subdivisé en deux (2). Il est passé d'un module de quarante et cinq (45) heures à deux (2) modules de trente (30) heures (soit 60 heures). Le premier est entièrement consacré à l'algorithmique et le second à la programmation.

Cette augmentation du volume horaire consacré à cet enseignement-apprentissage a autorisé une baisse du taux moyen d'échec de l'ordre de quatre-vingt-huit pour cent (88 %). En d'autres termes, le taux d'échec est passé de quatre-vingt-treize virgule deux pour cent (93,2 %) à quatre-vingt-quatre virgule cinq pour cent (84,5%). Cet écart de cinq points environ sur les trois dernières années est intéressant. Toujours est-il que l'échec reste important. Une réflexion plus profonde sur le processus enseignement-apprentissage de l'algorithmique à l'ENSET de Libreville s'impose donc pour tenter de remédier à cette situation.

Sur le plan traditionnel, l'enseignement-apprentissage de l'algorithmique et/ou d'un langage informatique s'appuie sur des problèmes à résoudre. L'enseignement-apprentissage de l'algorithmique et des langages informatiques impliquent alors chez l'apprenant deux activités cognitives : l'apprentissage du formalisme de l'algorithmique,

en lien avec un langage informatique, et une démarche de résolution de problème. De ce fait, l'apprenant est amené, d'une part, à apprendre le formalisme de l'algorithmique, puis celui du langage informatique, et d'autre part, à résoudre le problème posé par la tâche exigée. Cette double activité cognitive ne facilite pas chez les apprenants l'apprentissage en raison de la charge cognitive générée par cette double activité (Amadieu & Tricot, 2006).

Il est nécessaire d'indiquer que les interrogations sur l'enseignement-apprentissage de l'algorithmique et des langages informatiques actuellement effectués à l'ENSET de Libreville s'articulent autour des interrogations suivantes :

- Pourquoi et comment sont-ils enseignés ?
- Quelle est leur place dans le curriculum de formation des élèves professeurs de l'ENSET de Libreville ?
- Quels sont les savoirs portés par cet enseignement ?
- De quelle manière le processus enseignement-apprentissage se met en œuvre dans de telles situations ?
- Et enfin, comment envisager l'élaboration de nouvelles approches pour permettre la construction d'une pédagogique adaptée (Ginestié, 2008) ?

Dans la pratique, il est admis que les élèves apprennent mieux lorsqu'ils manipulent des équipements en situation réelle. Dans cette perspective, les enseignants proposent aux élèves des séances de travaux pratiques, avec du matériel souvent sophistiqué (ordinateurs, microscopes, etc.). En termes d'apprentissage, peut-on affirmer que cet apport cadre avec la réalité ? Le temps consacré à la manipulation, sans compter celui qui est nécessaire à la préparation, à la commande et la récupération du matériel, à la rédaction des consignes, est-il bien optimisé ?

L'étude du processus enseignement-apprentissage de l'algorithmique et des langages informatiques conduit inéluctablement à s'intéresser aux travaux portant sur l'usage de l'ordinateur dans l'enseignement (Technologies de l'Information et de la Communication pour l'Enseignement, ou TICE), en discutant d'un certain nombre de questions en liaison aux instruments informatiques (Bruillard, 2001 ; 2006 ; 2013).

Au regard de ce qui précède, cette étude présente, dans un premier temps, des apports théoriques sur le processus d'enseignement-apprentissage à partir des concepts de la didactique des sciences et de l'informatique, de la psychologie des apprentissages, dans le contexte d'un apprentissage des sciences et de l'informatique. Il y est développé dans une deuxième partie quelques usages de l'ordinateur dans l'enseignement, puis certaines approches des concepts d'algorithme et de résolution de problèmes dans l'enseignement-apprentissage de l'algorithmique, avant d'aborder les aspects liés aux outils de simulation d'algorithmes. Notre dispositif expérimental, les résultats auxquels nous sommes parvenus, une discussion sur ces résultats, une analyse comportementale de nos sujets ainsi que quelques perspectives ouvertes par cette étude sont présentés dans la dernière partie.

# 1. Apprentissage traditionnel et algorithmique

## 1.1. Enseignement-apprentissage de l'algorithmique à l'ENSET de Libreville

L'École Normale Supérieure de l'Enseignement technique (ENSET) est un établissement placé sous la tutelle des Ministères de l'Enseignement Supérieur et de l'enseignement technique et professionnel. Créée et organisée par l'ordonnance 81/72 du 30 décembre 1972, l'École Normale Supérieure de l'Enseignement Technique (ENSET) du Gabon a pour mission :

- la formation initiale et continue ainsi que le perfectionnement pédagogique des enseignants techniques des secteurs publics, parapublics et privés ;
- l'organisation en étroite collaboration avec les départements ministériels concernés et notamment le ministère de l'enseignement technique et professionnel, les travaux de recherches pédagogiques relatifs aux enseignements techniques ;
- l'établissement des contacts nécessaires avec les secteurs économiques concernés par les problèmes de formation des ouvriers qualifiés et des techniciens de tout niveau (Bekale Nze ; Ginestié & al., 2014).

L'admission au sein de l'ENSET se fait par voie de concours ouvert à tous les bacheliers scientifiques et technologiques. Au terme de leur formation d'une durée de trois (3) ans, cinq (5) ans ou huit (8) ans, les étudiants sont respectivement diplômés d'une licence, d'un master ou d'un doctorat dans plusieurs options des sciences et techniques industrielles ou du tertiaire (voir cursus de formation à l'ENSET en annexe 11).

Les enseignements à l'ENSET se présentent sous forme de modules. Ces modules sont regroupés en unités d'enseignement (UE). Dans le cadre de l'informatique, il y a trois principaux modules, le premier est consacré à la description et au fonctionnement des ordinateurs (Informatique 1), le second est l'algorithmique (Informatique 2) et enfin le troisième est totalement dédié à l'étude d'un langage de programmation (Informatique 3). L'enseignement-apprentissage de l'algorithmique et de la programmation, notre centre d'intérêt, s'effectue au deuxième et au troisième semestre.

### 1.1.1. Pourquoi enseigner l'algorithmique

Un constat populaire que d'ailleurs nous partageons, à propos de l'enseignement des mathématiques est que : « les mathématiques partagent quotidiennement notre

existence, mais cette omniprésence sans cesse croissante, est souvent occultée aux yeux du public qui ne voit que le produit fini ». Cette constatation concerne également les algorithmes dont on observe souvent les résultats que les principes fondamentaux. La présence d'algorithmes dans l'univers technologique au quotidien n'est plus à démontrer. En partant du simple automate jusqu'aux systèmes les plus complexes, les algorithmes organisent beaucoup nos gestes quotidiens. Cependant, leur présence n'est pas directement perceptible pour l'utilisateur qui assimile volontiers «la machine» à son mode de fonctionnement.

C'est à ce titre qu'il nous semble important de nous demander pourquoi enseigne-t-on l'algorithmique? La réponse à cette interrogation nous édifiera donc sur l'importance que revêt l'enseignement de l'algorithmique à tous les niveaux du système scolaire et pour la formation des formateurs de l'enseignement technique et professionnel.

La démarche algorithmique constitue depuis très longtemps un élément fondamental de l'activité mathématique. En maternelle, petite section et au primaire déjà, les programmes consacrent une rubrique au « développement de la pensée logique ». Il s'agit ici d'identifier des propriétés, de confronter, d'ordonner, d'organiser une action et de tirer les conséquences de son effet, d'identifier ou d'appliquer une règle, de codifier, de symboliser, etc., pour finalement permettre plus tard à l'apprenant d'enrichir les connaissances sur les formes, les mots, les éléments, les faits considérés, l'aptitude à classer et à maîtriser des connaissances en jeu. Au collège, les élèves étudient et surtout se servent également de certains algorithmes comme les algorithmes opératoires, l'algorithme des différences, l'algorithme Euclidien et les algorithmes en géométrie qui servent à la construction. Le programme suggère une formalisation en langage naturel. Dans certains établissements d'enseignement supérieur, l'algorithmique est une discipline transversale qu'il est important d'étudier avant d'apprendre un langage de programmation (informatique ou d'automate programmable).

Dans des établissements d'enseignement supérieur qui n'ont pas pour vocation la formation des analystes programmeurs, l'enseignement de l'informatique (algorithmique) doit permettre à l'apprenant de connaitre l'outil informatique, sa description et son fonctionnement, ses différentes utilisations dans la société, les différentes possibilités tant matérielles que logicielles, et permettre aux apprenants, au terme de leur formation, de proposer au moins des débuts de solution dans leur domaine d'activité sur des problèmes informatiques. Il est clairement mentionné dans la lettre de saisine du Conseil supérieur des programmes du 19 décembre 2014 que l'enseignement de l'algorithmique offre la possibilité d'assimiler des méthodes qui développent la pensée algorithmique, la logique programmée ainsi que des compétences dans la formulation et le traitement de l'information, les résolutions de problèmes et l'examen des productions. L'enseignement de l'algorithmique, offre la possibilité aux apprenants, à court terme d'être en mesure d'écrire des programmes. Cet objectif transféré dans les enseignements de spécialité permet à terme aux apprenants de

développer des programmes capables de piloter des machines automatisées, dont la finalité est commandée par le fonctionnement de telles machines.

L'algorithmique donne également aux enseignants la latitude de développer un enseignement orienté vers une pédagogie de projet, active et collaborative. Il est question de faire une initiation à l'algorithmique et à la programmation dans le but d'avoir la maitrise du langage de description d'algorithmes (LDA) et d'un langage de programmation informatique. L'emploi du langage de description d'algorithmes ou d'un langage de programmation est le moyen d'apprendre d'autres démarches d'investigation, de nouvelles méthodes de résolution de problèmes, de simulation ou également de modélisation. En effet, la maitrise de l'algorithmique ou d'un langage de programmation n'est pas la finalité d'un tel enseignement. Les problèmes ne sont pas résolus qu'en programmation. A l'école comme partout ailleurs, notre quotidien est agrémenté de problèmes à résoudre. Il est donc important de s'y préparer.

Selon la notification du Ministère de l'Éducation National de mars 2016 relative à l'enseignement des TIC, le programme du cycle quatre en mathématiques, il est mentionné que l'enseignement de l'algorithmique et de la programmation développe chez l'apprenant des compétences liées à la décomposition (l'analyse d'un problème compliqué, le découpage en sous-problèmes, en sous-tâches) ; à l'identification de schémas (reconnaitre des schémas, des configurations, des invariants, des répétitions, mettre en évidence des interactions) ; à la généralisation et d'abstractions (repérer les enchaînements logiques et les traduire en instructions conditionnelles, traduire les schémas récurrents en boucles, imaginer des procédés en relation avec des objets qui expriment le comportement attendu) ; enfin à la conception d'algorithme et de programme (produire des solutions modulaires à un problème précis, se resservir des algorithmes déjà programmés, programmer des instructions occasionnées par des événements, imaginer des algorithmes s'exécutant concomitamment).

Le constat de toutes ces orientations institutionnelles et officielles concernant l'enseignement-apprentissage de l'algorithmique et de la programmation nous incite à se poser des questions sur la manière de mettre en œuvre cet enseignement.

## 1.1.2. Comment enseigner l'algorithmique

Après avoir établi l'importance et les buts visés par l'enseignement de l'algorithmique en nous appuyant sur certaines recommandations officielles, il nous semble donc opportun de s'interroger sur les modalités d'enseignement-apprentissage de l'algorithmique.

En algorithmique, selon des extraits du programme officiel de 2016, les modalités de l'apprentissage peuvent être variées. En effet, le programme préconise de travailler individuellement ou en groupe, en salle informatique ou en salle banalisée, sur tablette ou sur ordinateur. Pour cet apprentissage trois types essentiels de démarche sont envisagés. Il s'agit de:

- une démarche de projet active et collaborative qui met en exergue la définition d'objectifs partagés et la distribution des tâches, les échanges entre élèves contributeurs du projet commun favorisant les échanges interdisciplinaires.

- une démarche de création : l'enseignant encourage les productions collégiales (programmes, applications, animations, etc.). Pendant ces productions collectives, les élèves développent leur autonomie, l'esprit de créativité et d'imagination, mais aussi le sens du travail collaboratif donc à se socialiser.

- une démarche interdisciplinaire qui facilite la réalisation de plusieurs activités de production numérique dans un contexte d'enseignements complémentaires.

A ces trois démarches, il est nécessaire d'ajouter une démarche de résolution de problème dans laquelle l'apprenant est seul, face à une situation problème développe l'autonomie, la logique, le transfert, la consultation de sources de références, le tâtonnement au sens de l'apprentissage par essai-erreur dans lequel l'apprenant procède aveuglément, en tâtonnant, et les erreurs s'excluent petit à petit pendant les prouesses successives, tandis que les essais utiles se raffermissent et se succèdent avec de plus en plus d'assurance (Reboul, 1980), et aussi le tâtonnement expérimental qui pour Célestin Freinet (1992) est basée sur la trace que laisse en l'individu l'action menée à bien devenu automatique, c'est le réflexe conditionné qui s'inscrit dans l'agissement matériel et automatique de l'individu...

A l'ENSET de Libreville, l'apprentissage de l'algorithmique et de la programmation peut se faire au cours de deux semestres consécutifs la même année ou sur deux années consécutives : un semestre consacré à l'algorithmique et le suivant à la programmation.

Le programme actuellement définit comme objectif général: « ... être capable d'écrire un algorithme ou un programme simple ». Ce programme privilégie essentiellement l'analyse d'un problème qui aboutit à la conception d'un algorithme, mais aussi la phase pratique de programmation effective.

Dans le cadre des changements préconisés, il est question de faire évoluer cet objectif en : « être capable d'écrire, de débuguer (corriger) et exécuter un algorithme ou un programme simple ». Cette évolution est motivée par l'introduction dans l'apprentissage d'un logiciel de simulation d'algorithme. Ainsi, l'accent est mis non seulement sur l'analyse du problème à résoudre en vue de concevoir l'algorithme ou le programme, mais aussi sur une étape de débogage. Cependant, il ne faut pas nécessairement insister sur une chronologie précise (algorithme, débogage puis programmation), ces trois phases avançant souvent en parallèle : le débogage comprend la phase d'essais-erreurs où l'élève construit petit à petit un algorithme ou un programme qui répond au problème qui lui est posé. De même, au moment de l'encodage du programme ou de l'algorithme sur l'interface de saisie, l'élève peut se retrouver confronté à une question algorithmique qu'il n'avait pas intégrée dans son analyse initiale.

En résumé, au regard de ce qui précède, il apparait inéluctablement que l'enseignement-apprentissage de l'algorithmique s'inscrit dans le cadre général de l'enseignement-apprentissage des sciences et particulièrement de l'informatique. Dans cette perspective, que peut-on alors mentionner sur l'algorithme et sur son enseignement-apprentissage.

# 1.2. Algorithme et enseignement-apprentissage de l'algorithmique

## 1.2.1. Algorithme : historique et essai de définition

Avant la naissance des ordinateurs et de l'informatique, plusieurs générations de savants se sont succédé pour inventer des théories mathématiques et logiques. Ces découvertes vont impacter considérablement le milieu informatique parce qu'elles seront à l'origine des programmes et des logiciels développés aujourd'hui. La discipline fondamentale qui est l'algorithmique que tout informaticien, du moins tout programmeur, se doit de connaître est particulièrement identifiée (Hernert, 2002).

Le mot algorithme est un dérivé du nom d'un illustre savant Perse (actuellement l'Iran) *Muhammad Ibn Musa Al Khawarizmi* qui a vécu au neuvième siècle de l'ère chrétienne, lorsque Abbasside Al-Ma'mun était calife. Il était l'un des adhérents de l'un des centres de recherche de pointe de l'époque, le *Bayt Al-Hikma* « la Maison de la sagesse ». Grand mathématicien de son époque, al-Khawarizmi a participé à la traduction d'importants manuscrits scientifiques grecs et indiens. C'est sous sa plume qu'est sortie ce qui est toujours considéré comme le premier manuel d'algèbre le *Hisab al-jabr w'al-muqabala*. Ce manuel évoque parmi beaucoup d'autres sujets la résolution des équations du premier et du second degré. Le terme encore d'usage de nos jours « algèbre » émane directement du titre de cet ouvrage qui a été traduit en latin par Robert de Chester en 1145 sous la forme : *liber algebrae et almucabola*. Cet ouvrage révolutionna les mathématiques car il évoqua pour la première fois un système de résolution des problèmes : la méthode des algorithmes. Cette méthode des algorithmes consiste en la simplification des données complexes en données élémentaires, ou d'équations complexes en équations simples. Sa pratique nécessite donc une méthodologie, un esprit logique et énormément de rigueur dans les notations. En fait, on peut simplement dire qu'un algorithme est une succession d'instructions élémentaires. Nous sommes confrontés quotidiennement aux algorithmes, lorsque par exemple nous cuisinons, la recette que nous utilisons est un algorithme. En effet, la recette est une suite d'instructions simples (Hernert, 2002).

Mais au-delà de cette constatation générale, les algorithmes sont présents de façon volontaire ou pas dans bon nombres de sciences, des sciences humaines aux sciences dites exactes. Cependant, c'est les mathématiques et l'information qui ont fait le plus connaitre les algorithmes. Qu'en est-il alors de la notion d'algorithme dans ce domaine ?

## 1.2.2. Algorithme : approche mathématique et informatique

L'informatique considère un algorithme comme une méthode commune servant à résoudre un ensemble de problèmes. Un algorithme est dit « correct » quand il offre la solution au problème posé. Des critères comme la durée de calcul, le remplissage de la mémoire RAM, la précision des résultats obtenus, etc., permettent de mesurer l'efficacité d'un algorithme. Les ordinateurs utilisés pour exécuter ces algorithmes ne sont pas rapides à l'infini. En effet, le temps mis par une machine pour effectuer une tâche reste une ressource limitée, malgré les progrès constants effectués dans ce domaine. Un algorithme est dit « performant » s'il utilise rationnellement les ressources mises à sa disposition, c'est-à-dire le temps CPU (Central Processing Unit), la mémoire et la quantité d'électricité consommée. On réalise souvent une analyse de la complexité algorithmique afin d'estimer l'évolution du temps de calcul nécessaire pour qu'un algorithme évolue à son terme, en fonction du nombre de donnés à traiter.

L'algorithmique est donc l'étude des algorithmes. Cette discipline, proche des mathématiques et de l'informatique, s'intéresse à la création, la description et l'analyse des algorithmes (Knuth, 1981).

Le concept d'algorithme est inséparable de l'évolution des mathématiques. Les algorithmes sont présents actuellement dans beaucoup de domaines, qu'ils soient dérivés plus ou moins directement des mathématiques comme la recherche opérationnelle ou les techniques de programmation, ou bien qu'au contraire ces domaines cherchent à s'en rapprocher pour mettre à profit la rigueur que peuvent offrir des outils formels, comme cela se voit dans différentes filières de la psychologie et aussi de la pédagogie. Cet usage transdisciplinaire soulève la question de savoir si toutes ces disciplines donnent aux algorithmes le même sens, la même définition. L'éclaircissement de ce point nous paraît essentiel à la compréhension de la place qu'occupe ce concept dans les mathématiques, l'informatique et dans les sciences humaines.

Les origines de l'informatique remontent aux mathématiques de l'antiquité, à travers deux principales approches : l'algorithmique, qui organise la notion de «calcul», et la logique, qui met en forme la notion de «démonstration». Ces deux approches sont déjà très visibles dans la science grecque: Archimède et Diophante «calculent» la surface sous une parabole et les solutions de systèmes d'équations en nombres entiers, tandis qu'Euclide met en évidence les axiomes de la géométrie élémentaire, et Aristote conceptualise la logique propositionnelle. Il est à noter que ces deux courants originels représentent toujours la base de l'informatique contemporaine.

Jusqu'au XIXe siècle, de grands mathématiciens, comme Newton, Leibniz, Euler ou Gauss, découvrent des méthodes inédites de calcul numérique et symbolique. Ces méthodes doivent être mises en œuvre par un calculateur humain, cependant la rigueur desdites méthodes dresse déjà les prémices à l'époque des fondements de l'informatique. En parallèle, au tour du XXe siècle, le courant axiomatique prit son essor et accrocha de nombreuses branches des mathématiques ayant des interrogations

méthodologiques qui sont à l'origine d'une nouvelle discipline : la logique mathématique. C'est ce courant qui découvrira la théorie générale de la calculabilité (Post, Turing, Kleene, Church) et plusieurs autres théories de la démonstration (Gentzen, Herbrand, Heyting). Ces théories représentent le deuxième appui de l'informatique : dès qu'il sera nécessaire de formaliser la notion d'algorithme, de définir des langages de programmation claire servant à écrire des algorithmes en utilisant une expression non-ambiguë, de vérifier la cohérence de langages et de programmes, elles s'avéreront particulièrement précieuses. Quelle définition les mathématiques et l'informatique donnent-elles alors d'un algorithme ?

Les mathématiques et l'informatique considèrent en première approche qu'un algorithme est synonyme de procédé de calcul et de méthode de résolution. Les opérations arithmétiques les plus élémentaires sont des algorithmes ; quand on a appris l'algorithme de l'addition, on sait faire toutes les additions.

Par algorithme, on doit percevoir une méthode systématique de calcul. Le sens médiéval est celui de méthode pour effectuer les quatre opérations arithmétiques de base (formalisée par Al-Khawarizmi au IXe siècle), mais il s'étend rapidement à tout procédé de calcul.

Pour Knuth (1981), c'est un ensemble fini de règles donnant une série d'opérations afin de résoudre certains problèmes spécifiques.

Kitajima (2004) quant à lui souligne le fait que l'algorithme est un ensemble d'opérations nécessaires pour résoudre un problème en décrivant les étapes qui permettent d'atteindre le résultat.

En informatique, on donne la précision qu'il s'agit d'une succession d'opérations effectuées selon un ordre précis afin d'aboutir à un résultat en un nombre fini d'étapes. Si pour les mathématiques, les algorithmes sont des procédés de calcul, l'informatique complète le fait que ledit procédé de calcul doit se faire en une succession d'étapes qui mène à la résolution d'un problème. Le but visé en informatique étant de traduire un algorithme dans un langage de programmation informatique.

Historiquement, une proposition sur l'existence d'un algorithme, pour un problème donné, était accompagnée de sa description complète ; le caractère algorithmique pouvait immédiatement être prouvé par la simple mise en œuvre des règles fournies. Cela peut permettre d'expliquer pourquoi le concept de procédé de calcul a mis tant de temps à s'individualiser, à se séparer d'une description matérielle qui en cachait la généralité.

A un moment donné, les mathématiciens se sont intéressés au fait qu'un algorithme pouvait résoudre un certain nombre de problèmes. Ce développement a été lié majoritairement à la recherche de procédés encore plus puissant, c'est-à-dire de techniques qui sont en mesure de traiter sous une forme unique des classes de problèmes encore plus vastes.

On est conduit ainsi au problème qu'a soulevé Leibnitz, la construction d'un procédé de résolution pour un problème de mathématique arbitraire. Cependant, certains problèmes ont pu trouver des solutions de façon générale. Il a donc été possible de résoudre des problèmes pour des cas particuliers, mais des solutions n'ont pas été obtenues pour le cas général.

Devant l'impossibilité de résoudre certains problèmes mathématiques, il était donc question pour certains de démontrer mathématiquement cette impossibilité. Cette situation a donc fait apparaitre simultanément plusieurs nécessités :

- d'une part, le concept de procédé de résolution s'est dégagé, abstraction faite d'un problème précis ou d'une solution précise ;
- d'autre part, la notion intuitive d'algorithme devrait être reformulée, de manière à en faire un objet mathématique rigoureux mais commode, permettant un calcul formel.

Il n'est pas possible de limiter notre analyse de l'approche mathématique du concept d'algorithme qui l'évoque en tant que procédés de calcul sans parler de la notion de calculabilité. La calculabilité est alors une notion totalement abstraite. Elle ne tient compte ni du problème précis, ni des contraintes nécessaires à la réalisation effective du calcul donné. Il nous semble parfaitement illustratif de montrer que l'on peut formaliser la notion de calculabilité sous la forme d'une machine à calculer idéale, c'est l'idée originelle qui a permis l'élaboration de la machine de Turing.

Au delà de toutes les idées mathématiques et informatiques qui concrétisent la notion d'algorithme, nous pensons que l'algorithme est d'abord une idée, une pensée qui prend naissance et se forme dans l'esprit de son concepteur. Dès lors, il est alors important de connaitre l'approche la psychologie à propos de l'algorithme.

## 1.2.3. Algorithme : le point de vue de la psychologique

La définition que donne de l'algorithme la psychologie est fort proche de celle de la pédagogie. En effet selon Benedetto (2013) il s'agit d'une suite de règles dont on a la certitude qu'elles nous conduiront à la solution du problème.

Dans la perspective de Vergnaud (2002), il s'agit d'une procédure permettant de résoudre en un nombre fini de pas, tout problème d'une classe donnée à l'avance ou de montrer qu'il n'a pas de solution. Il ajoute que l'algorithme est aussi une organisation invariante de l'activité pour une certaine classe de problèmes.

Si l'on se place d'abord dans la perspective d'un processus enseignement-apprentissage, l'apprenant, spontanément, essaye de construire des systèmes de règles, d'élaborer des algorithmes (pas ceux qui sont portés par les systèmes automatisés et/ou l'informatique) qui permettent de résoudre des problèmes. Les enseignants observent souvent chez les apprenants une tendance à construire leur propre procédé de

résolution ou de rendre systématique l'utilisation de certaines règles. On peut en conclure que si les apprenants essaient de construire des algorithmes, ces derniers sont souvent faux ou incomplets, car ils s'appuient sur des procédés faux.

En milieu professionnel, il est observé souvent l'existence de telles constructions systématiques, spontanées pour l'accomplissement d'une tâche. Si l'intérêt du psychologue se fonde encore une fois sur l'existence de comportements spontanés de type algorithmique, on garde jusqu'à un certain point l'ambigüité du statut de ces comportements spontanés. Cependant, ces exemples montrent la valeur d'une attitude méthodique, qui oriente systématiquement l'analyse du travail vers la recherche d'éléments permettant de définir des comportements algorithmiques quand cela est possible. Dans ce cadre où la complexité des problèmes est très élevée, seul une analyse complète de la tâche permet de construire un algorithme.

Selon Vergnaud (1968), il y'aurait bien deux propriétés des algorithmes : la propriété d'effectivité, déjà connue, et la propriété de nécessité. La propriété d'effectivité est rendue possible en raison du rapport de nécessité : « […] dans le cas de l'algorithme, quand on met en œuvre ces règles d'action, de prise d'information et de contrôle, on aboutit en faisant un nombre déterminé de pas (étapes), en raison des rapports de nécessité entre les propriétés des actions et les propriétés des objets ». Pour qu'un algorithme s'achève après avoir effectué un nombre fini de pas, il faut (condition nécessaire) qu'il y ait cette relation. Cette question de la nécessité renvoie alors à la question de la rationalité de l'action : « si l'on se contentait du critère de la finitude qu'en serait-il de la nécessité, de la rationalité ? » Les algorithmes prennent leur sens dans les situations nécessaires et productives. Ils permettent d'obtenir des réussites à coup sûr en produisant des effets sur le monde.

Si l'on veut introduire un peu de rationalité dans les procédures transmises (et plus ou moins imposées) par la culture, une manière à leur prêter l'idée d'algorithme, il convient à tout le moins d'expliquer les relations entre les propriétés des actions et les propriétés des objets : pourquoi on fait ceci ? Cela ? Comment ça marche ?

Dans les règles d'action, de prise d'informations et de contrôle n'aboutissant pas en un nombre fini de pas, on peut se contenter de dire que les sujets ne suivent pas correctement les algorithmes. Mais la question est alors de savoir ce qu'ils font, Vergnaud (2001) nous enseigne qu'ils transforment le fruit de ce qu'ils ont appris en schèmes personnels. Cet auteur précise que les algorithmes sont des schèmes, mais que tous les schèmes ne sont pas des algorithmes. Les schèmes ne conduisent pas à coup sûr à la réussite. Ils auraient une marge d'incertitude. L'algorithme serait caractérisé par l'effectivité qui est la propriété d'aboutir à la réussite en un nombre fini de pas. Le distinguant ainsi du schème.

Un autre intérêt de la psychologie vis-à-vis des algorithmes se situe dans un des cadres d'origine de la théorie des algorithmes appliquée aux sciences humaines. Dans ce cadre, certains auteurs ont introduit à côté de la définition proprement dite, une

typologie. Elle distingue deux catégories d'algorithmes, les algorithmes de résolution, et les algorithmes de reconnaissance ou d'identification. Cette distinction peut de prime abord être considérée comme artificielle puisqu'elle semble se fonder uniquement sur une distinction de but, alors qu'il s'agit toujours d'algorithme et que seul le problème à résoudre est changé. En fait, de nombreux arguments justifient l'existence de cette typologie. Il est primordial de rappeler l'importance des algorithmes pour la pédagogie dont la finalité n'est pas uniquement la transmission à tout prix des seuls procédés de résolution, mais aussi la transmission de la compréhension de leur logique et des conditions de leurs utilisations. On peut opposer à cette attitude la réalisation d'un algorithme pour une machine dont l'unique but recherché est d'aboutir à la solution dans les meilleures conditions. Dans la résolution de problèmes, la principale difficulté ne consiste pas à réaliser les opérations prévues par l'algorithme, mais de découvrir par l'analyse le type dont relève le problème à résoudre et d'identifier les conditions d'applicabilité de l'algorithme. En effet, la réalisation d'un algorithme de résolution tient compte uniquement de la logique propre du problème. Par contre l'établissement d'un algorithme de reconnaissance constitue une tâche spéciale. La structure de ses opérations ne dépend nullement du contenu concret des objets sur lesquels ils s'appliquent, mais du schéma de relations qui lient entre eux les différents caractères qui définissent un objet (Landa, 1961).

La théorie du reflet est aussi à évoquer. Elle est fondée sur le fait qu'il existe une correspondance entre l'activité psychique et l'activité pratique de l'homme, l'activité psychique étant l'activité pratique extérieure transformée comme un processus de transition du reflété au reflet. Pour Vygotsky, les processus psychique se forment par enrichissement de l'extérieur vers l'intérieur. Landa (1966) parle de règles selon lesquelles doit se dérouler l'activité psychique de l'homme, ce sont les règles de l'activité psychique exigées. En fait, pour cette théorie décrire, le monde matériel et les relations entre les objets qui le composent c'est décrire des opérations psychiques mise en œuvre. Ceci existe et ne peut être possible uniquement parce qu'il y a reflet c'est-à-dire qu'il y a un mécanisme de passage de l'extérieur vers l'intérieur : ce qui permet de rendre compte du rôle joué par les algorithmes dans le processus enseignement-apprentissage.

En résumé, il est observé qu'il y a successivement une forme matérielle (forme de départ), l'objet de l'action est donné à l'apprenant sous la forme d'objets réels ou de modèle, schèmes ou dessins, puis la forme extérieure verbale et enfin la forme intellectuelle. Dans le cadre d'un apprentissage, un algorithme représente donc la forme de départ sous une forme matérialisée. C'est donc cette forme matérialisée dont se sert la pédagogie afin que l'algorithme permette aux apprenants de résoudre des problèmes.

## 1.2.4. Algorithme : les contraintes pédagogiques

Depuis la découverte des algorithmes et leurs formalisations, les sciences humaines à l'instar des mathématiques se sont approprié le concept. En effet, certains chercheurs ont orienté leurs travaux dans l'application des algorithmes en pédagogie. Plusieurs de

leur production nous renseignent sur le sens que ces sciences donnent à la notion d'algorithme et aussi éventuellement à l'usage qu'elles en font.

Landa (1966) parle d'une suite d'opérations tellement rigoureuse qu'elle permet de résoudre tous les problèmes d'une classe donnée. Cet auteur définit également des algorithmes comme une succession d'opérations définies en nombre fini ayant pour objet de résoudre une classe de problèmes donnée.

De ces deux définitions il ressort entre autre deux aspects essentiels : un algorithme permet de résoudre « tous les problèmes » d'une classe donnée, pour la première définition ; et pour la seconde un algorithme permet de trouver des solutions à « une classe » de problèmes donnée. Si on se réfère à ces définitions, il existe donc deux types d'algorithmes, ceux qui résolvent tous les problèmes et ceux qui ne résolvent pas tous les problèmes, donc qu'une classe de problèmes. Certains chercheurs parlent d' « algorithmes absolus » au sens de la théorie mathématique et d' « algorithmes aboutissants ».

Certaines recherches en mathématiques permettent de dégager une notion plus réaliste de la calculabilité introduite plus haut. Mais les théories mathématiques traitant de la calculabilité ne constituent pas une approche au calcul pratique. Un algorithme aboutissant est un algorithme permettant d'arriver à un résultat compte tenu des contraintes. La notion d'algorithme aboutissant peut paraitre paradoxale au premier abord puisqu'une des propriétés des algorithmes est l'effectivité, c'est-à-dire précisément la propriété d'être une solution certaine et *a priori* d'une classe de problème donnée. Cependant, le paradoxe disparait aussitôt si l'on se situe dans la hiérarchie des niveaux de définition des algorithmes, car dans un cas, on se situe au niveau de la théorie mathématique, l'effectivité est une propriété des algorithmes absolus. Le fait qu'on puisse introduire la précision d' « algorithme aboutissant » revient bien à réintroduire les contraintes particulières d'un système donné. Si le système qui doit effectuer le calcul est un opérateur humain, pour qu'il y ait algorithme aboutissant il faut tenir compte des contraintes caractérisant cet opérateur.

Concernant l'identification des objets sur lesquels s'applique l'algorithme, l'analyse fait remarquer un mélange entre les propriétés des algorithmes et les propriétés des systèmes qui vont appliquer ces algorithmes. Cependant il est constaté que dès la résolution des différents problèmes permettant de caractériser un algorithme aboutissant, ces définitions sont tout à fait semblables et les propriétés doivent être identiques. On peut énumérer entre autres les propriétés suivantes:

- la définition : être entièrement décrit aussi bien dans les différentes étapes élémentaires que dans l'enchaînement de ces étapes ;

- des instructions : c'est un ensemble de consignes de taille finie, même s'il comporte un très grand nombre de consignes élémentaires ;

- l'effectivité : être effectif est ici pris au sens d'une efficacité a priori (Vergnaud, 1968) ;

- puissant : permettre de résoudre une classe de problèmes plus ou moins large.

Ces propriétés décrivent bien les algorithmes, indépendamment du niveau de définition auquel on se situe.

En définitive, la pédagogie ne peut dissocier les propriétés des algorithmes des limitations auxquelles ils doivent se plier pour être vérifiées. Pour l'opérateur humain, ces problèmes sont difficiles à poser dans le cadre de systèmes formels permettant de savoir s'ils sont résolubles ou pas, s'il existe un algorithme ou pas. Cependant, ce qui importe à la pédagogie c'est la règle (l'algorithme). Elle doit concevoir et enseigner des algorithmes qui permettent à l'apprenant de résoudre des problèmes. Vu ainsi, la pédagogie, afin de permettre aux apprenants d'utiliser des algorithmes, doit veiller qu'ils soient entièrement décrit aussi bien dans les différentes étapes élémentaires que dans l'enchainement de ces étapes, qu'ils soient constitués d'un ensemble d'instructions (consignes) de taille finie, qu'ils soient efficaces et enfin qu'ils soient puissants. Pour la pédagogie, les algorithmes utilisés dans les différents enseignement-apprentissage doivent permettre de résoudre tous les problèmes pour lesquels la règle a été conçue ; pour peu qu'elle ne souffre pas de problème d'applicabilité. Dans un usage pédagogique, donc en situation de résolution de problème, la règle ne peut se contenter de résoudre certains problèmes mais pas d'autres. Si elle est établie pour une classe de problèmes, elle doit pouvoir tous les résoudre au sein de cette classe. De plus, la pédagogie doit également veiller à enseigner non seulement la règle, mais aussi les conditions d'application de ladite règle sans lesquelles la règle, donc l'algorithme, ne peut être appliqué. En effet, il est important de comprendre que pour la pédagogie, les algorithmes n'ont pas pour unique but la transmission à tout prix des seuls procédés de résolution, mais aussi la transmission de la compréhension de leur logique et des conditions de leur utilisation.

Finalement, que l'on y porte un regard mathématique, informatique, psychologique ou pédagogique, les algorithmes sont des procédés de résolution de problèmes. Il nous semble alors important après la découverte du procédé de comprendre les deux notions qui l'accompagnent : le problème et la résolution de problèmes.

## 1.3. Problèmes et résolution de problèmes

Il est toujours important de rappeler que l'algorithmique est l'étude des algorithmes. Un algorithme est de façon générale un procédé qui sert à résoudre un problème, plus précisément, c'est la description d'une solution à un problème dans un formalisme particulier. Il semble donc judicieux de s'interroger sur la signification, non seulement du mot « problème », mais aussi de l'expression « résoudre un problème ». Ainsi, afin de mieux cerner le concept de résolution de problèmes dans le contexte de notre travail, il est opportun de l'envisager sous l'angle psychologique et enfin sous l'approche

didactique et pédagogique. Avant toutes choses, interrogeons-nous d'abord sur le sens général à donner au mot « problème » et à l'expression « résolution de problèmes ».

Dans son acception la plus courante, un problème est une situation à travers laquelle un obstacle empêche de progresser, d'évoluer, d'avancer ou d'atteindre son but ou son objectif. Un problème naît de l'écart entre la situation actuelle et celle souhaitée, ou lorsqu'il y a anormalité. On peut aussi parler de disparité ressentie entre une situation perçue et une situation désirée.

Certains domaines d'activités présentent aussi de façon particulière ce qu'ils entendent par « problème ». En effet, en industrie, un problème est simplement défini comme l'occurrence d'événements qui perturbent le fonctionnement habituel d'un système robotique ou productique. Dans le champ de l'analyse fonctionnelle, le problème est un écart (une différence) observable entre un résultat attendu et le résultat constaté. L'informatique théorique quant à elle considère des problèmes comme étant des questions auxquelles un ordinateur peut répondre. Enfin, un problème physiologique résulte d'un dysfonctionnement ou anormalité d'un composant ou système d'un organisme vivant.

Certains auteurs parlent d'une situation stimulus requérant une certaine réponse de la part d'un sujet qui ne possède à cet instant ni totalement, ni partiellement cette réponse. La définition du problème couvre trois principales assertions :

- il y a problème par rapport à un sujet, précisément par rapport à l'état de ce sujet ;
- la réponse doit correspondre aux caractéristiques de la situation, elle doit être conforme à certains critères ;
- La réponse n'est pas automatique, elle ne doit pas être rattachée implicitement aux éléments qui déterminent la situation problème de façon réflexe ou à la suite d'apprentissages.

Indépendamment du domaine d'activité et de la définition que nous avons du problème, nous nous s'accordons pour dire que la résolution de problèmes est un exercice qui vise à préciser la situation préoccupante en la définissant de façon rigoureuse et précise, pas du tout confuse et à la traiter dans le but de déterminer une solution.

Que l'on soit dans la phase de constatation du problème ou celle qui consiste à résoudre le problème constaté, il faut initialement une opération mentale qui permet, face à une situation donnée, de réaliser qu'il y a un problème, puis après entreprendre la démarche cognitive permettant de le résoudre. Finalement le problème et la résolution de problème sont des concepts qu'ils convient d'examiner sous l'aspect psychologique.

### 1.3.1. Résolution de problèmes : les apports de la psychologie

En psychologie, un problème est souvent défini comme une tâche à accomplir selon des conditions précises et pour laquelle on ne connaît pas de solution ou de méthode systématique de résolution: on sait ou pas clairement quel est le but à atteindre, on connaît le contexte (lois ou règles à respecter, outils ou transformations licites, ...), mais on ne connait pas formellement la procédure à suivre. La résolution du problème consiste alors autant à inventer (ou définir) la méthode à appliquer (les étapes à suivre) qu'à les mettre en œuvre. Bien évidemment, dès que le problème est résolu, la (ou une) méthode est connue. Le problème n'en est alors plus un.

Richard (2005) suggère que pour résoudre un problème, il faut au préalable avoir les connaissances précises du contexte spécifique de la situation-problème, de produire les raisonnements nécessaires afin de pouvoir identifier les actions possibles et les ordonner. Ainsi, il faut donc avoir un ensemble de connaissances qui soient liées à la situation problème afin de produire le raisonnement dont on a besoin afin de pouvoir résoudre le problème formulé par la tâche prescrite. Ainsi, la résolution de problème est une activité mentale conduisant à la compréhension d'une situation qui pose problème au sujet et aboutissant à une action. Autrement dit, il est question de mettre en œuvre des procédures connues, sous la forme d'une construction nouvelle car intégrant les éléments de la situation, et produite par raisonnement (Mayer, 2008). Notre réflexion sur le processus enseignement-apprentissage de l'algorithmique voire des langages informatiques conduit donc à s'interroger sur ces mécanismes du raisonnement et de compréhension qui aboutissent *in fine* à l'élaboration de la solution au problème.

Quand un problème est posé, on sait que c'est un problème, mais dans la vie on ne sait pas toujours ce qui va être un problème : souvent on pense savoir exécuter une tâche et à un moment donné cela se révèle être un problème. La notion de problème se situe donc dans le contexte de l'action qui découle de la connaissance acquise. La transformation d'une connaissance en une action est une opération très complexe surtout lorsque c'est la première fois. En effet, l'apprenant qui passe à l'action n'est toujours pas certain de l'action qui découle de la connaissance, sauf dans le cas limite ou on reproduit toujours exactement la même action dans la même situation. De plus, il faut passer des connaissances en mémoire aux décisions d'action et à leur organisation temporelle. II faut pour cela tenir compte du cadre, des notions préalablement acquises dont certains peuvent être satisfaits d'autre non. Un traitement est nécessaire pour appliquer les procédures connues au contexte de la situation. Nous distinguons deux types de situations : celles qui mettent en jeu seulement des activités d'exécution et celles qui requièrent des activités de résolution de problème (Richard, 2004).

Richard (2004) reprend Hoc (1984) en distinguant soigneusement les activités d'exécution et les activités de résolution de problèmes. On ne peut faire cette distinction en considérant seulement les tâches : certaines tâches sont des problèmes pour certains

sujets et sont des situations d'exécution pour d'autres. II faut donc faire intervenir le couple sujet-tâche et définir cette distinction par les processus en jeu.

La situation de résolution de problèmes découle du fait que les connaissances dont on a besoin pour élaborer une procédure acceptable et utilisable face à la situation n'existent pas en mémoire: soit les connaissances qui concernent les représentations mentales, soit les connaissances qui permettent de faire les inférences nécessaires à l'ordonnancement des actions. Ce qui détermine un problème ou une situation d'exécution, ce n'est pas seulement la situation, c'est la relation existante entre la tâche et les compétences du sujet. II convient donc de préciser comment une situation est ou devient un problème pour le sujet et comment une situation qui était un problème devient une situation d'exécution.

On peut se trouver dans une situation problématique de deux façons :

- On n'a pas en mémoire les connaissances permettant de décider des actions à faire dans la situation. En d'autres termes, aucun schéma d'action correspondant au problème n'est activé. Nous nous trouvons donc face à une situation qui nécessite la construction d'un espace de recherche et trouver la solution au sein de cet espace. Dans ce cas, la situation est bien perçue comme un problème par le sujet (Richard, 2004).

- Il y a en mémoire des connaissances applicables, ces connaissances ont été appliquées et elles ont échoué. On se trouve alors dans une situation que nous qualifierons d'incident. Au départ, la situation n'apparaissait pas problématique, elle l'est devenue parce que les procédures connues ne conduisent pas au résultat escompté. II faut, dans ce cas, remettre en cause la représentation que l'on a de la situation. La situation devient un problème parce qu'elle a des contraintes qui ne sont pas celles des situations dans lesquelles les procédures connues sont applicables. II convient donc d'identifier ces contraintes et construire de nouveaux opérateurs (Richard, 2004).

Du point de vue de la conscience que le sujet peut avoir d'être dans une situation problématique, on peut avoir plusieurs cas de figure : le sujet peut avoir pleinement conscience qu'il est confronté à un problème, il peut aussi ne pas avoir conscience qu'il va être devant un problème. Dans d'autres cas de figures, l'apprenant peut aussi ne pas être certain que la connaissance qu'il applique est adaptée pour le contexte de la situation : il sait qu'elle est adaptée pour d'autres contextes, il ne sait pas vraiment si elle l'est pour celui-ci, mais comme il n'a pas d'autre connaissance qui fournisse une solution, il utilise celle-là. C'est typique dans le cas d'utilisation d'analogies (Richard, 2004).

Après la description de ces situations problématiques, il est utile de se demander comment une situation-problème se transforme en une situation d'exécution. Une situation qui est un problème devient une situation d'exécution quand l'apprenant a

construit, pour le contexte spécifique de cette situation, les représentations mentales des actions qui rendent possible la réalisation des sous-buts et que par ailleurs il a les heuristiques et les connaissances nécessaires pour déterminer les sous-buts. Pour qu'une situation-problème puisse devenir une situation d'exécution, il faut donc que l'apprenant ait en mémoire les connaissances sur les représentations mentales de l'action conforme à la situation. Si c'est une situation non familière, la définition des actions à réaliser se fait par un processus d'opérationnalisation qui utilise ces connaissances (Richard, 2004).

On ne saurait parler de la résolution de problème d'un point de vu de la psychologie sans parler du rôle de l'erreur. L'analyse des erreurs a montré que les comportements apparemment chaotiques observés en résolution de problème sont en fait très systématiques et résultent soit de l'application de procédures erronées soit des représentations inappropriées mais qui peuvent se comprendre. C'est la situation observée notamment dans les erreurs commises chez les élèves lorsqu'ils effectuent les soustractions à retenues. Brown (1988) et Van Lehn (1990) ont montré que ces erreurs étaient dues à des procédures imparfaites et à des tentatives de réajustement quand ces procédures conduisent à des impasses : ils ont pu trouver les règles qui sont à l'origine de la plupart de ces erreurs. Sander (1997) a montré que les enfants n'ayant pas encore appris la soustraction à retenue faisaient des erreurs également systématiques qui se reproduisent à peu près intégralement à intervalles de quelques semaines : ces erreurs peuvent dans leur quasi-totalité être engendrées à partir de deux modèles intuitifs familiers aux enfants de cet âge, le modèle de retrait d'une partie d'un tout et le modèle de la distance dans un parcours entre deux bornes.

En définitive, quel que soit l'angle dans lequel il est abordé en psychologie, comprendre le processus cognitif de la résolution de problème permet de mettre en lumière, d'analyser puis d'expliquer les difficultés d'apprentissage des apprenants. Ces analyses servent inéluctablement en didactique pour mettre en place des expérimentations qui débouchent sur des solutions que la pédagogie met en œuvre dans les salles de classe afin d'améliorer le processus enseignement-apprentissage.

### 1.3.2. Résolution de problèmes : le regard de la didactique et de la pédagogie

En pédagogie, on utilise souvent l'expression « la situation problème » pour parler du problème. En effet, dans les pédagogies dites actives, en l'occurrence, la pédagogie par situation-problème, la pédagogie par projet, la différenciation pédagogique…, l'enseignant donne à l'élève un problème à résoudre. Généralement, la démarche ou la règle à utiliser pour résoudre le problème a été préalablement enseignée ou est à découvrir présentement. Pour la didactique, le problème posé doit avoir une solution connue. Son but est donc d'enseigner la règle (l'algorithme) de résolution du problème.

Vu sous cet angle, l'activité didactique s'intéressera plus à la résolution de problèmes qu'au problème en lui-même.

D'Hainaut (2000) pense que résoudre un problème implique des structures de connaissances nouvelles au niveau de la situation et met en jeu des concepts et des opérations appris antérieurement mais qui sont mis en œuvre d'une nouvelle manière. Cet auteur estime que pour résoudre un problème, il faut d'abord qu'il y'ait acquisition de connaissances, ce sont ces connaissances qui' plus tard, seront structurées d'une nouvelle manière et utilisées pour construire la solution.

La résolution de problèmes a de tout temps été une activité primordiale en pédagogie, mais le rôle fonctionnel de cette activité est apprécié différemment selon qu'on l'observe dans le cadre d'une pédagogie dite traditionnelle ou dans celui de pédagogies dites actives.

La pédagogie traditionnelle se caractérise par le fait qu'elle établit distinctement deux moments dans l'enseignement-apprentissage. Le premier concerne l'élaboration du travail à effectuer par l'élève. Celui-ci est réglé selon deux phases successives : une phase d'acquisition (la leçon, le cours magistral ou *ex-cathedra*), une phase d'utilisation des connaissances (l'exercice d'application encore appelé travaux dirigés). La plupart des manuels scolaires traditionnels mettent en évidence cette distinction fondamentale entre leçon et les exercices sur la leçon. Le deuxième moment, est en définitive une justification du premier, il porte sur le bilan des apprentissages à l'école, c'est-à-dire qu'il établit clairement une distinction entre question de cours et problème (Brousseau, 1998). Quels objectifs visent-on alors à travers le problème : tester l'acquisition des connaissances, opérationnaliser ces dernières sous forme de capacités ?

A l'inverse, dans la perception des pédagogies actives, le problème est le point de départ de l'apprentissage et non comme l'aboutissement de celui-ci. Deux principes généraux sont ainsi admis. Le premier est que toute personne impliquée dans un problème, donc confrontée à une situation pour laquelle elle ne dispose pas de réponse immédiate, déploie un certain nombre de démarches pour le résoudre. Le deuxième est de faire en sorte que le travail effectué doit laisser des traces, de telle sorte que ces traces puissent servir à décanter plus tard une situation identique ou similaire. L'apprenant pourra répondre de façon plus directe, sans avoir à reproduire toutes les démarches ayant conduit à la solution initiale (Dessus, 2010). Ainsi, dans l'apprentissage, on peut se servir de la pédagogie du problème pour entretenir implicitement la distinction traditionnelle entre connaissances et capacités, le problème étant perçu ici comme motif, comme mobile, pour la recherche et l'acquisition de connaissances. Le principe pédagogique de base est que, d'une part, la situation problème enthousiasme l'activité de l'élève, et que, d'autre part, la recherche de la solution impose que soient réuni et inclu un certain nombre de données non encore retenues par l'élève. De plus, une tout autre vision peut nous amener à dire que la résolution du problème n'est pas la raison d'être de l'apprentissage des connaissances,

elle constitue l'apprentissage même. La différenciation entre connaissances et capacités, entre savoir et utiliser, est proscrite. Toute activité, motrice ou symbolique, est tributaire de « savoirs faire », c'est-à-dire de règles qui guident les actions générales afin de produire des réponses aux différentes circonstances auxquelles on peut faire face. Les apprentissages permettent d'agrandir le répertoire de règles dont dispose un apprenant, de telle sorte qu'il soit capable de répondre à un nombre important de sollicitations variées. Ces règles ou façons de faire apprises sont des capacités. Il n'est pas nécessaire qu'elles soient formulables verbalement pour être acquises en tant que telles (D'Hainaut, 2000).

En définitive, nous pensons qu'il y a problème lorsqu'aucune des règles ou façons de faire dont dispose une personne n'est plus directement applicable à la situation. L'activité mise en œuvre par l'apprenant consiste alors à élaborer des hypothèses en s'appuyant sur l'ensemble ou les sous-ensembles des règles déjà acquises et parmi lesquelles peuvent se trouver des règles correspondantes à de telles hypothèses, c'est-à-dire des algorithmes, des stratégies, des principes heuristiques, etc. Toutes ces notions qui constituent des éléments fondamentaux de l'enseignement-apprentissage des sciences en général et de l'enseignement-apprentissage de l'informatique en particulier.

# 1.4. Enseignement-apprentissage des sciences et de l'informatique

## 1.4.1. Enseignement-apprentissage des sciences et de la technologie

Depuis plusieurs décennies, la problématique de l'enseignement-apprentissage des sciences et de la technologie occupe de nombreux scientifiques (De Boer, 1991). La présence constante de la science et de la technologie dans notre quotidien nous a amené à hisser comme enjeux majeurs de l'avenir de nos sociétés la formation scientifique et technologique. L'accent a été mis sur l'enseignement scientifique général, dans le but de faire acquérir aux élèves les connaissances basiques en sciences, non pas simplement pour former des scientifiques, mais pour rendre la science accessible à la majorité de la population. C'est à ce titre que Legendre (1994) affirme que : « la vulgarisation de la science dont le but essentiel est l'acquisition d'une culture scientifique de base pour tous, comporte la double dimension sociale et individuelle. Sur le plan social, il s'agit de permettre à la société d'exercer un certain contrôle sur les progrès scientifiques et techniques, et de participer davantage aux choix technologiques fondamentaux, alors que sur le plan individuel, on souhaite conduire les individus à se servir autant que possible au quotidien des connaissances scientifiques ». La formation aux sciences participe aussi au développement de l'esprit critique, du raisonnement logique, ainsi que de la pensée conceptuelle. C'est donc une formation fondamentale qui elle-même renvoie globalement au rôle de l'enseignement des sciences sur le développement

individuel et social. Toutefois, en dépit de la place privilégiée qu'occupe l'enseignement scientifique dans les lycées et collèges, on constate, ici et là dans le monde, un échec assez conséquent (Walberg, 1991). En effet, on constate partout un faible niveau de réussite, un manque d'intérêt pour la science en général et une attitude assez peu scientifique à l'égard des phénomènes. Ainsi, en dépit de tentatives répétées pour améliorer les programmes et pour assurer une formation en harmonie avec les exigences de la société contemporaine, l'enseignement des sciences n'a pas en définitive réussi à atteindre les buts fixés.

Le Gabon n'échappe pas à cette tendance. En effet, la loi d'orientation générale de l'éducation de la formation et de la recherche proposée par les états généraux de l'éducation de la recherche et de l'adéquation formation emploi de mai 2010 dans son titre 1, au chapitre 2 consacré aux missions de l'enseignement de la formation et de la recherche stipule à l'article 5 que les curricula, les offres de formation et les équipements doivent permettre selon le niveau l'appropriation des connaissances et des compétences en matières de sciences et technologies ainsi que des technologies de l'information et de la communication. Dans la même loi, le titre 3 au chapitre consacré aux établissements d'enseignement et de formation préconise l'ouverture des lycées d'émergences scientifique (LES) afin d'accueillir les élèves après la troisième ayant des moyennes supérieures ou égales 12/20 dans les disciplines scientifiques pour les préparer aux différents baccalauréats scientifiques. Les conclusions de ces travaux recommandent au pré-primaire et au primaire de susciter un éveil scientifique et technologique, dans le secondaire il est question de développer les formations scientifiques techniques et professionnelle et de moderniser le plateau technique en équipement de pointe.

Comme nous le constatons, l'enseignement-apprentissage des sciences et de la technologie au Gabon, malgré son organisation et sa présence à tous niveaux ne donne pas, comme partout ailleurs, les résultats escomptés ainsi qu'en témoigne ces nombreuses recommandations et propositions d'organisation. Une reformulation à tous les niveaux permettra certainement de revoir le dispositif surtout en ce qui nous concerne dans l'enseignement-apprentissage des sciences et des technologies.

Notons que l'apprentissage de l'élève n'est pas uniquement influencé par le contenu de l'enseignement, mais également par la manière dont le professeur enseigne. Or, les stratégies pédagogiques adoptées par les enseignants sont en général le reflet d'une observation à la fois empirique et inductive de la science qui est constamment éloigné de l'image contemporaine de la science (Larochelle et Desautels, 1992; Linder, 1992; Ruggieri, Tarsitani et Vicentini, 1993). Cette perception empiriste considère que les modèles scientifiques ou les concepts théoriques se dégagent peu à peu des faits qui sont eux-mêmes révélés à l'observation sans aucun présupposé. La méthode scientifique qui consiste à induire des lois et des théories sur la base d'observation de faits s'uniformise. Il convient donc de dire qu'à travers cette vision, l'élaboration progressive des modèles théoriques de la science est négligée. De plus, le rôle des théories est

insignifiant car la méthode scientifique donne à l'observation un statut illusoire, en cachant son caractère social et construit. Les modèles scientifiques sont des outils dont le but est de donner des représentations sélectives et symboliques d'un phénomène capables de générer l'action. L'attrait des modèles scientifiques réside dans les possibilités d'actions qu'ils suggèrent et les nouvelles expérimentations qu'ils suscitent car ces modèles ne sont pas découverts mais bâtis pour cet usage (Fourez, 1992). C'est ainsi que le rôle de l'expérimentation ne se réduit pas à démontrer, mais à opérationnaliser des modèles pour en tester les limites. Pourtant, au cours des activités de laboratoire, la démarche scientifique se réduit souvent pour l'apprenant à une manipulation ayant pour objet d'appliquer, les unes à la suite des autres une suite d'étapes définies à l'avance. Ces activités sont rarement organisées en une démarche de résolution de problèmes. Par conséquent, les élèves concentrent toute leur attention sur le résultat à obtenir que sur le processus mis en œuvre. Cette manière de mener les activités de laboratoire fait que l'élève confonde plus tard le mécanisme de réalisation et la règle de présentation d'une démarche de recherche (Bybee; Powell et Ellis, 1991).

Au Gabon, dans le cadre de l'enseignement des sciences physiques dans l'enseignement général et des essais et mesures dans l'enseignement technique et professionnel, l'enseignant met à la disposition des apprenants le schéma de montage expérimental et une description point par point de la manipulation que les élèves doivent effectuer. La seule difficulté étant à ce niveau de reconnaitre sur le schéma les symboles des différents équipements (voltmètre, ampèremètre, inductance, capacitance, résistance…). Dès que le montage est effectué par l'apprenant, l'enseignant vient le contrôler et demande de commencer les relevés. En faisant varier un des paramètres, l'apprenant de façon mécanique se contente de lire les valeurs qui s'affichent sur les appareils de mesure (qui quelque fois sont numériques, donc à lecture directe). La seconde étape consiste souvent à construire sur un papier millimétré la courbe des relevés effectués. A cet étape, on constate souvent deux attitudes : soit l'élève sait ce qu'il doit obtenir (c'est souvent le cas de ceux qui reprennent la classe) donc il oriente volontairement les mesures pour avoir une belle courbe, et quand il s'agit d'un élève novice, les données produisent une courbe un peu éloignée du résultat escompté dont il a des difficultés à en tirer quoi que ce soit. Lorsque l'enseignant organise une telle manipulation en laboratoire, c'est pour permettre à élève d'établir une relation logique entre les différents paramètres qu'il manipule à travers le montage expérimental. Mais tel que c'est fait, les élèves pensent que le but est de tracer la courbe, ils ne perçoivent pas qu'à travers l'allure de la courbe, ils doivent faire une analyse qui doit aboutir à l'établissement d'une relation entre les paramètres qu'ils ont manipulés.

L'enseignement des sciences et de la technologie n'est pas en marge de certains courants de pensée. En effet, cela s'observe dans les exemples choisis et dans les références à l'histoire que l'on rencontre parfois dans les manuels, dans certaines images que l'on véhicule concernant la méthode scientifique ainsi que dans les postures et dans les stratégies qu'adoptent les enseignants, dans l'essence même du langage scientifique et dans le type de dialogue qui s'instaure entre le formateur et l'apprenant

dans les enseignements scientifiques (Linder, 1992). Fourez (1992) met en évidence l'idée selon laquelle l'évolution des sciences engendre inéluctablement le progrès. Une telle réflexion rend le progrès uniquement tributaire d'une seule dimension et établit une hiérarchie entre le savoir de l'expert et les autres types de savoirs. Il constate aussi la tendance qui consiste à présenter les sciences comme neutres et basées sur des faits observés indépendamment de tout projet particulier. Cette tendance, que l'on trouve souvent dans l'enseignement, mettant l'accent sur le caractère désintéressé de la science et s'opposant à son aspect utilitaire peut conduire à un éloignement artificiel entre science, technologie et société.

A l'ENSET de Libreville, il est demandé aux chefs de départements dans le cadre des projets de fin de cycle de mettre en place avec les étudiants des projets qui sont non seulement adapté au niveau technologique du pays, mais aussi, dans la mesure du possible qui soient utiles aux besoins de l'établissement. C'est ainsi dans une option comme le génie électrique, un projet d'automatisation et sécurisation d'un portail est utile par rapport à la construction d'une maquette d'un train à grande vitesse (TGV). Ce genre de projets scientifiques et technologiques ne valorise pas le caractère désintéressé puisqu'il met l'accent sur une production en adéquation avec le niveau technologique du pays et réalise aussi un produit qui peut être utile et utilisé soit par l'institution de formation soit partout ailleurs dans le pays.

On se rend bien compte que les critères qui permettent de choisir des concepts, des modèles ou des théories, ne sont pas nécessairement établis sur la base logique ou rationnelle (Fourez et Lambert, 1989), mais en s'appuyant sur des décisions ou des besoins liées à un contexte social, historique et quotidien. Ces orientations sociales données à la science et à la technologie sont en définitive faiblement évoqués dans les programmes et manuels scolaires. On estime que ces aspects vont engendrer des changements assez radicaux dans le choix et dans l'organisation mêmes des contenus traditionnellement enseignés (Hodson, 1992).

Il convient de souligner que la nature même du jargon scientifique ne facilite pas la perception des concepts. En effet, dans l'apprentissage des sciences l'élève doit mémoriser énormément les concepts et les principes dont les significations ne lui sont pas souvent connues. La nature même du vocabulaire scientifique pose des exigences particulières pour la mémoire. Il faut se souvenir de beaucoup de mots, leur signification et leur orthographe que l'on découvre. Walker (1989) établit que le caractère ésotérique du vocabulaire scientifique représente un obstacle majeur à la compréhension des élèves. Les travaux en psychologie cognitive ont permis d'établir une distinction entre la mémoire sémantique, qui est de nature conceptuelle, et la mémoire lexicale qui correspond à la mémoire de la forme des mots. Selon Lieury (1992), la difficulté des mots scientifiques a pour origine leur complexité à la fois lexicale et sémantique. L'élève peut mémoriser des termes scientifiques, c'est-à-dire qu'il peut retenir le mot, sans pour autant savoir sa définition ni savoir l'écrire correctement (Meyerson, Ford, Jones & Ward, 1991 ; Stepans, 1991). Par ailleurs, on peut également trouver des homonymies,

dans ce sens que le terme peut disposer des significations très différentes dans le langage courant et dans le langage scientifique. Mémoriser, ce n'est pas donc simplement reproduire par cœur, c'est surtout comprendre, c'est-à-dire faire des inférences et générer des réponses à de nouvelles questions. Pour cela, il importe d'organiser correctement les concepts en mémoire sémantique et plus cette mémoire possède des informations, plus il est facile d'y ajouter de nouvelles (Dupin et Joshua, 1989). Ainsi, pour que les termes scientifiques soient faciles à comprendre, il faut au préalable s'assurer que les apprenants possèdent des connaissances requises permettant l'assimilation puis les aider à relier divers concepts entre eux et à les hiérarchiser. Le recours à des cartes ou des réseaux conceptuels, l'utilisation d'analogies ou des métaphores constituent des outils pédagogiques et didactiques intéressants (Dupin et Joshua, 1989).

Une autre source de difficulté réside dans le caractère formel et abstrait des connaissances scientifiques et technologiques et dans le rôle central qu'y joue la modélisation mathématique. Quelques recherches ont été faites sur l'étude de la maîtrise par les apprenants des schèmes opératoires formels jugés prépondérants pour la compréhension des principes, des concepts ainsi que les méthodes de la science (Lawson, 1985 ; Padilla, 1991). De ces recherches, il ressort que beaucoup d'élèves n'utilisent pas les outils intellectuels et les processus d'abstraction nécessaires à l'apprentissage des sciences et de la technologie. Ce qui signifie que face à un problème en science, les élèves ne se servent pas des modèles, théorèmes et autres outils théoriques mathématiques appris en cours pour résoudre le problème, ils s'appuient en priorité sur des exemples et modèles concrets. Doit-on rendre la science moins abstraite, plus accessible à l'apprenant? Faut-il user en priorité des méthodes d'enseignement des sciences plus concrètes, plus descriptives en s'appuyant systématiquement sur des outils technologiques? Certains ont tenté de concevoir des programmes susceptibles de stimuler la pensée critique et des outils logiques et mathématiques en rapport avec la pensée formelle. Mais comment peut-on développer ces habiletés sans tenir compte des contenus sur lesquels elles portent (Glaser, 1987)? L'enseignement des mathématiques et des sciences ne peut-il pas favoriser le développement du raisonnement logique et des capacités d'abstraction se rapportant à la pensée opératoire formelle au sens que décrit Piaget? Une entrave à l'évolution du raisonnement logique est le fait d'insister sur la valeur numérique des résultats plutôt que sur la qualité du raisonnement à l'origine de ces résultats. L'élève apprend de cette façon à appliquer des formules et à effectuer des calculs, sans pour autant travailler les habiletés de raisonnement qui permettent de catégoriser les problèmes et de réduire l'éventail des opérations possibles. Piaget (1972) a d'ailleurs attiré l'attention sur le danger de l'introduction prématurée en science de la quantification métrique au détriment des processus de raisonnement plus qualitatifs.

Toutefois, signalons que le caractère formel ou abstrait des concepts enseignés n'est pas seul à l'origine de leurs difficultés de compréhension mais, peut-être davantage, le manque de lien entre formalisme et savoirs spontanés. Les savoirs spontanés mis en

œuvre par l'apprenant en rapport avec les divers contenus qui lui sont enseignés sont souvent en conflit ou en contradiction avec les modèles enseignés et sont ainsi à l'origine d'une part importante des difficultés associées à l'apprentissage de la science (Clement, Brown et Zietsman, 1989 ; Driver, 1989 ; Giordan et De Vecchi, 1987 ; Giordan, 1991). Nous remarquons que ces représentations ne sont presque pas corrigées par l'enseignement. Elles altèrent souvent les notions enseignées. Quelque fois, il arrive que ces deux types de représentations, bien qu'elles soient conflictuelles, puissent coexister dans l'esprit de l'élève sans que ce dernier n'en soit conscient. Au quotidien, ces représentations prennent généralement le dessus sur les connaissances scolaires. L'élève éprouve ainsi des difficultés à s'approprier le savoir scientifique qui lui est enseigné car il est incapable de le comprendre en s'appuyant sur ses connaissances antérieures ou ses conceptions spontanées. Cet intérêt à l'égard des savoirs que l'élève a acquis avant est lié à une approche constructiviste de la connaissance. En effet, pour les tenants de cette conception, l'apprentissage est un processus actif qui n'a pas pour but d'emmagasiner les informations édictées par l'enseignant, mais à les traiter et à les changer grâce aux connaissances déjà acquises (Giordan, 1991; Grayson, 1991).

En définitive, pour apprendre la signification des concepts scientifiques et technologiques, l'apprenant doit la reconstruire en s'adossant sur les connaissances et représentations initiales. Tout apprentissage dépend ainsi des représentations mobilisées par l'apprenant pour interpréter les informations. Il ne consiste pas à concaténer des connaissances nouvelles et antérieures, mais implique un changement de conceptions.

En résumé, si nous admettons que dans les sciences et technologies l'apprentissage ne constitue nullement une accumulation de notions anciennes et nouvelles, mais elle se caractérise par un changement de paradigme chez l'apprenant, en est-il de même pour l'enseignement-apprentissage de l'informatique et des TIC ?

## 1.4.2. Enseignement-apprentissage de l'informatique

Les technologies de l'information et de la communication (TIC) et l'informatique représentent aujourd'hui un axe majeur de préoccupation de notre civilisation moderne du fait de son haut niveau de technicité et de l'engouement qu'ils suscitent auprès de nos contemporains. L'école ne pouvait rester en marge de cette tendance et nombreux sont les états qui investissent dans ce secteur dans le but de rendre leurs systèmes éducatifs plus efficace et plus efficient. En effet, tout le monde s'accorde sur la nécessité de faire une place aux TIC en éducation et à l'outil informatique qui les porte. Cette mouvance semble incontournable. De ce fait, de nombreux gouvernements ont orienté leurs politiques publiques vers des programmes d'intégration des TIC en milieu éducatif dans la mesure où l'apprentissage de l'informatique s'appuie sur l'usage des TIC.

Le concept de technologies de l'information et de la communication (TIC) en éducation renvoie aux équipements technologiques numériques qui peuvent servir d'outils pédagogiques. Nous citons entre autre, les ordinateurs, serveurs, caméras numériques, caméras vidéo numériques, numériseurs, projecteurs, lecteurs de cédéroms, lecteurs de DVD, graveurs, imprimantes, modems, logiciels, etc. Intégrer ces outils dans l'éducation signifie une cohabitation féconde et pacifique entre les TIC et ses prédécesseurs dans la chaîne éducative afin de produire un enseignement remarquable et un excellent apprentissage. Dias (1999), défend à ce propos le caractère utile de l'intégration des technologies quand elles servent de manière continue afin de promouvoir les objectifs du programme, conduisant par la même occasion les apprenants vers des apprentissages qui ont un sens. La publication de l'OCDE (2015) intitulée « Students, computers and learning. Making the connection » montre comment l'utilisation des TIC à l'école dépend non seulement de leur disponibilité, mais aussi des politiques relatives aux enseignants et aux programmes d'études, les curricula.

En revanche, il est aussi important de souligner que l'intégration ne doit pas simplement consister à introduire du matériel informatique à l'école sans modifier les pratiques pédagogiques. En parlant d'intégration des TIC, Mangenot (2000) soutient que l'intégration de l'outil informatique est tributaire de son efficacité au service des apprentissages. Dans le même ordre d'idées, Hérold & Ginestié (2017) sur la question de l'efficacité des outils informatiques, montrent à travers les résultats de leur recherche que cette efficacité est relative à certaines disciplines et/ou avec certains étudiants. En effet, nous pensons qu'il ne s'agit pas de mettre plus d'ordinateurs, ni de les utiliser tout le temps, mais de rendre simplement leur présence et leur utilisation efficace pour faciliter l'apprentissage. Ces ordinateurs peuvent être utilisés par les apprenants ou par les enseignants soit pour mieux faire ce qu'ils font déjà, soit pour faire d'autres choses, les deux approches étant pertinentes au plan pédagogique.

Depover (1996) propose deux manières d'intégrer les TIC en milieu éducatif. Une approche descendante qui propose la situation selon laquelle la décision d'intégration dérive d'une autorité politique, et une approche ascendante par laquelle l'initiative

d'intégration émane des enseignants eux-mêmes. Nous estimons que les deux approches proposées ici sont complémentaires dans la mesure où les décisions politiques ne peuvent que légaliser la démarche du corps enseignant. L'institution politique étant le pouvoir organisateur duquel dépend l'institution scolaire.

Ainsi décrite, cette intégration des TIC à l'école suppose de revisiter les méthodes traditionnelles d'enseignement centrées sur l'enseignant dont il est le détenteur des connaissances qu'il transmet. Face à cette révolution dans les pratiques pédagogiques occasionnée par l'utilisation des TIC dans l'enseignement, révolution qui place désormais l'apprenant au centre en le rendant acteur de son apprentissage transforme profondément les métiers d'élève et d'enseignant en modifiant les rapports au savoir de chacun. Dans un tel apprentissage, la mise en situations problèmes en groupe et en équipe permet la construction du savoir. La résolution du problème exige que l'apprenant confronte sa solution à celles de ses homologues (El Mouthi et al, 2013). L'apprenant se sert essentiellement des ressources issues du travail collaboratif (apprentissage collaboratif), et son activité réflexive devient plus manifeste dans les activités personnelles et collectives. Connor, Karmokar et Whittigton (2015) considèrent, qu'un des éléments majeurs des approches pédagogiques comme la résolution de problème, est le fait que l'étudiant est actif dans son apprentissage, ce d'autant, si, de plus, cet apprentissage est porté par un outil informatique.

On peut alors s'interroger sur la proposition du système scolaire, ainsi que sur l'ampleur de la reconnaissance des TIC et de l'informatique dans nos écoles. Graduellement, les prescriptions institutionnelles au sujet de l'usage scolaire des technologies de l'information et de la communication ont changé. Cependant, trois expressions sont perceptibles de manière récurrente dans les discours officiels : un outil d'enseignement ; un ensemble d'outils disciplinaires ou transversaux et enfin un domaine d'enseignement nouveau.

De nouveaux domaines d'enseignement ont été constitués dès les années soixante (60) pour les disciplines technologiques. Finalement l'informatique n'a été considérée que comme un ensemble d'outils et de techniques sans enjeux épistémologiques et a plutôt fait l'objet de recommandations que de prescriptions. Pour l'école élémentaire, les orientations officielles de 1995, l'utilisation des technologies de l'information et de la communication n'occupe pas une place indispensable, en contradiction avec celles de 1985. Dans l'enseignement-apprentissage des TIC, l'enseignant familiarise l'élève avec l'utilisation de l'ordinateur qu'il met au service des disciplines et dont il fait appréhender les différentes possibilités. L'informatique est réduite au cycle des approfondissements. Quelques années plus tard, il est question de « maniement des ordinateurs », ce qui est manifestement un net recul de l'informatique et des TIC. Cette approche réduit à la seule pratique l'enseignement-apprentissage des TIC. Ainsi, c'est dans les options technologiques et de documentation que se produit la découverte explicite de l'informatique et des technologies de l'information et de la communication.

Il apparait donc indéniable que l'apprenant doit avoir des compétences sous forme de savoir-faire opératoires : l'accent porte surtout ici sur des éléments techniques. Toutefois, la précaution est prise dans les prescriptions officielles d'indiquer qu'il s'agit bien de faire acquérir aux élèves certaines compétences. Cependant, de nombreuse études menées aussi bien avec des élèves de seconde qu'avec des enseignants en formation initiale, corroborent que la plupart n'ont pas acquis les compétences adéquates octroyant l'autonomie dans l'exécution de leur tâche avec l'ordinateur et ne sont pas proche d'intégrer la culture informatique dans leur quotidien (Bruillard, 2000). Afin de transmettre les savoirs et les savoir-faire nécessaires, on est alors en droit de se demander quels moyens didactiques peuvent être déployés dans le processus enseignement-apprentissage de l'informatique ainsi que des TIC.

# 1.5. Didactique des sciences et didactique de l'informatique

La didactique des sciences et celle de l'informatique plus particulièrement sont au centre de notre recherche, tant les questions qui sont posées dans l'étude sont des questions de didactique et questionnent l'enseignement de l'informatique en général et le processus enseignement-apprentissage de l'algorithmique en particulier.

## 1.5.1. Didactique des sciences

Une réflexion sur les disciplines n'est pas indispensable lorsque nous avons pour objectif de transmettre des savoirs à une élite. En revanche, lorsqu'on s'oriente vers une éducation des masses, alors cette réflexion devient déterminante afin de construire des outils offrant à tous les mêmes possibilités d'accès aux savoirs (Meirieu, 2004). Selon Meirieu (1995) : « la didactique est constituée par l'ensemble des procédés, méthodes et techniques qui ont pour but l'enseignement de connaissances déterminées ». Pour Virgnioux (2009) : « la didactique est la discipline qui étudie ce qui est, dans les activités d'enseignement et d'apprentissage, spécifique des contenus ». Dans ces deux perceptions, nous dirons que la didactique s'attache et s'applique au contenu d'une discipline. La majorité des didacticiens mentionnent un « triangle didactique » dont les angles sont constitués par « le savoir », « l'apprenant » et « le formateur » ; en réalité, l'épistémologie de référence de la discipline considérée, la psychologie cognitive et les contraintes de la situation de formation sont les axes de réflexion (Houssaye, 1988). Dans le pôle épistémologique, on est confronté à des concepts comme celui de « transposition didactique », qui désigne le changement que subit un « savoir-savant » pour évoluer en objet d'enseignement (Chevallard, 1985). Les psychologues quant à eux parlent de « représentation mentale » pour désigner une construction psychologique de courte durée et qui cadre avec une explication circonstancielle du contexte auquel le sujet fait face (Richard, 2004). La manière de traiter l'information est aussi désigné par « opération mentale » (Hoc, 1984). Enfin, au sujet de la situation de formation, le concept de « contrat didactique », désigne les règles implicites qui régissent la relation entre

l'enseignant et l'apprenant. Le contrat didactique se différencie du contrat pédagogique à cause du fait que le contrat pédagogique ambitionne d'expliciter plusieurs objectifs et contraintes scolaires pour l'apprenant (Brousseau, 1986). Finalement, on parlera de la didactique d'une discipline pour désigner la science qui étudie, au sein de cette discipline, les phénomènes d'enseignement, les circonstances de la transmission des usages propre à une organisation ainsi que la manière dont l'acquisition de connaissances par un apprenant s'opère (Astolfi et Develay, 1992).

De ce qui précède, on peut déduire que l'évolution des didactiques, l'affermissement de leurs recherches a pour conséquence une meilleure efficacité de l'enseignement. De façon usuelle, les expressions « didactique des langues », « didactique des mathématiques », « didactique de la mécanique », « didactique des sciences », etc., peuvent être perçues comme étant une utilisation de techniques et de méthodes d'enseignement particulières dont dispose chaque discipline. En admettant que les techniques dépendent directement des contenus à enseigner, elles seront donc différentes selon les matières. Ainsi, dans les sciences économiques par exemple, on s'appuiera sur les techniques d'études de cas, alors que les sciences physiques privilégieront la démarche expérimentale (Vergnaud, 1994). Comment alors expliquer le fait que les didactiques soient de nos jours la source de polémiques si importantes ? En effet, l'origine de ces controverses est la parution de différentes publications critiques du « Dictionnaire des concepts fondamentaux des Didactiques » coordonné par Yves Reuter en 2010. Il s'agit des rapports entre les didactiques et la didactique, les écarts dans l'emploi du terme « didactique », l'éventualité d'une science didactique et enfin le bénéfice de mettre en place une didactique générale (Schneuwly, 2014 ; Chevallard, 2014 ; Reuter, 2014 ; Biagioli, 2014 ; Orange, 2014). Sans pour autant nourrir toutes ces polémiques, nous admettrons que la didactique des sciences s'occupe des savoirs scientifiques et technologiques à transmettre en impliquant leur évolution.

Dans la didactique des sciences, le recours à l'épistémologie revêt un caractère prépondérant. Il est souvent utile de s'appuyer sur une approche historique pour cerner efficacement les difficultés de l'acquisition du savoir, par l'identification des obstacles propres à sa constitution. L'épistémologie joue alors un rôle critique face aux idées usuelles sur les faits, les concepts, les lois et les théories scientifiques. Alors, on est en droit de se demander quelles sont aujourd'hui les idées capitales sur lesquelles repose la didactique des sciences ? La didactique des sciences véhicule trois principaux concepts autour desquels elle semble s'appuyer:

- le concept de représentation, provenant de la psychologie donc ayant une référence épistémologique, c'est Jean-Pierre Astolfi et Michel Develay en 1989 qui constatent son usage important en didactique des sciences ;

- le concept de transposition didactique, importé de la didactique des mathématiques par Yves Chevallard, cette idée vise à mélanger les processus de reformulation et de dogmatisation du savoir enseigné ;

- le concept d'« objectif-obstacle » défendu par Jean-Louis Martinand. Cette idée semble recueillir l'unanimité du fait qu'elle souhaite établir la prise de décision éducative et la mise en œuvre pédagogique entre la connaissance des évolutions historiques, des sens scientifiques et des problèmes psychologiques.

La didactique des sciences actuelles a un champ plus vaste qui s'étend au-delà du repérage des représentations, ou même des débats autour des didactiques, elle doit s'intéresser à la question essentielle qui est comment penser et organiser l'apprentissage des sciences ?

Les structurations ainsi que les réflexions qui sont entreprises sur l'apprentissage des sciences doivent être faites conjointement avec celles menées sur la formation des enseignants. En effet, l'un des axes important de cette formation passe par l'insertion dans le cursus de formation des enseignants des contenus de haut niveau. Cette formation ne doit nullement s'attacher plus ou moins exclusivement aux contenus qui feront l'objet de l'enseignement lui-même. Il est, en effet, indispensable que des enseignants soient confrontés, au cours de leur formation, à un savoir de niveau scientifique élevé, en constitution, jouxtant la recherche universitaire. Ce contact est indispensable pour l'intérêt personnel porté sur la discipline, pour comprendre aussi la naissance d'un savoir, les difficultés de sa constitution, son historicité, etc. En revanche, il est également fondamental que les enseignants travaillent sur les savoirs qu'ils auront à enseigner, qu'ils en maitrisent les enjeux, identifient ce qui peut être source d'entrave pour les élèves, imaginent des dispositifs didactiques, prennent en compte les différentes stratégies d'apprentissage qui peuvent exister, etc. Ce travail est tout aussi passionnant que le contact avec la recherche fondamentale, en réalité il est lui-même «recherche » et l'on peut ainsi accéder à de hauts niveaux de talent indispensables pour le succès de l'entreprise scolaire.

C'est dans cette optique que nous souhaitons inscrire notre travail sur le processus enseignement-apprentissage de l'algorithmique. En effet, tenter de comprendre l'échec nous amènera à analyser non seulement celui qui est en échec : l'apprenant, ce sur quoi il a échoué : le cours d'algorithmique, mais aussi comment l'enseignement est dispensé : le processus enseignement-apprentissage. Ainsi, nous allons regarder le fond et la forme de certains exercices qui sont proposés aux apprenants (informatique, algorithmique), puis tenter de comprendre comment l'apprenant apprend avant de résoudre un problème, processus cognitif mise en œuvre par ce dernier (psychologie cognitive), mais aussi le processus enseignement-apprentissage, dans la perspective aussi bien de l'apprenant que de l'enseignant (sciences de l'éducation, didactique), afin de comprendre puis de résoudre une situation de difficulté d'apprentissage qui conduit les apprenants vers un important taux d'échec. En définitive, nous nous engageons là indéniablement dans un travail de didactique de l'informatique.

## 1.5.2. Didactique de l'informatique

L'intérêt porté à l'informatique en milieu scolaire apparaît en France à l'entame des années soixante-dix (1970) avec une orientation particulière. A cette époque, c'est la démarche informatique, que certain qualifieront de démarche « algorithmique, organisatrice et modélisante » qui est alors jugée fondamental (Baron, 1989).

La didactique de l'informatique, pour sa part, apparaît dans les années quatre-vingt (1980). Pendant cette période, en France ainsi que dans plusieurs autres pays, les enseignements d'informatique sont dispensés dans les lycées. Puis autour de la même période, on observe progressivement un désintérêt de l'informatique comme objet de savoir, on se focalise de préférence sur ce qu'on va désormais désigner par l'expression « l'outil informatique ». Dans le discours officiel, cette expression désigne l'ensemble des outils logiciels qui se sont répandus à cette époque, le changement est plus qu'évident dans le sens. Dans le cadre de l'enseignement à l'école, l'informatique perd son statut de nom commun pour devenir un adjectif. Compte tenu de la disparition du cours d'informatique proprement dit, il devient alors difficile de faire des recherches en didactique de l'informatique.

Concernant la programmation et l'algorithmique, ils connaissent un regain d'intérêt. Dans l'enseignement primaire, la circulation de langages comme Scratch qui offrent un environnement de programmation par script ressemblant beaucoup à la construction d'un puzzle, semble se vulgariser. Au secondaire, l'enseignement de l'informatique est instauré, particulièrement en lycée où il est optionnel, dans les cours de mathématiques puis d'informatique (Nijimbere, 2015).

L'instauration de cet enseignement conduit inévitablement à mener une réflexion sur la didactique de l'informatique qui, elle-même, est fondamentalement attachée à la question de la reconnaissance de l'informatique comme une science à part entière.

Concrètement, une science se caractérise par les problèmes qu'elle pose, les résultats obtenus par les chercheurs qui se réclamant de ladite science, sont reconnus par des groupes scientifiques et des institutions qui en garantissent la validité. En somme, Georges-Louis Baron (2016) pense que les équipes de recherche qui planchent sur les enjeux didactiques de l'informatique sont prolifiques. Cette communauté se renouvelle et n'est pas organisée autour d'une théorie ou d'un modèle théorique privilégié, elle s'organise en travaillant sur un objet dont la définition n'est pas encore proprement établie. On peut alors se demander quelles sont les bornes du champ? Dans la communauté, certaines équipes s'intéressent principalement à l'enseignement de l'algorithmique, la pédagogie qu'il est possible de mettre en œuvre avec des instruments informatisés, ou encore les enjeux sociaux. Ce large éventail de chercheurs issus de domaines variés constitue indéniablement une source vivifiante.

Hormis cette question de la science informatique, l'instauration de l'informatique comme discipline scolaire et universitaire soulève aussi la question de la didactique de

l'informatique. Une des particularités de l'informatique est de faire accomplir une action à une machine qui n'a pas été mise au point pour réaliser une catégorie de tâches prédéfinie. C'est le programme qui renseigne sur les actions que la machine doit effectuer. A l'origine, les programmes ont été développés pour transcrire efficacement des algorithmes, au sens de suite d'opérations à effectuer sur un ensemble de données afin d'obtenir les résultats d'une certaine classe. C'est ainsi que l'algorithmique qui se définit comme la science de la conception d'algorithmes aura un rôle prépondérant dans l'édification de l'informatique comme discipline universitaire.

La didactique de l'informatique a vu le jour au cours de la deuxième moitié des années 1970, lorsque se sont interrogés des universitaires, dans le cadre de l'AFCET (Association Française pour la Cybernétique Économique et Technique), sur les moyens à déployer pour enseigner efficacement la programmation. Lors des séquences d'enseignement dans les classes à orientation informatique pendant les dix années qui ont suivi, des problèmes en rapports avec les didactiques sont identifiés de façon récurrente. En effet, on s'aperçoit très rapidement qu'il existe d'importantes difficultés dans l'apprentissage de la programmation. On constate en fait que les apprenants ont du mal à comprendre non seulement la nécessité de mener une phase d'analyse préalable à la programmation en spécifiant clairement les algorithmes à la base du programme, mais aussi le sens des codes du langage qu'ils utilisent (Rogalski & Samurçay, 1986, Rogalski 1990). En 1991, Christian Orange a proposé de prendre en compte la notion de pratiques sociales de référence suggérée par Jean-Louis Martinand et de projeter l'informatique dans les classes en mettant les apprenants face à des situation-problèmes qu'ils doivent résoudre (Orange, 1991).

Dans le cadre de notre recherche, à l'instar de Georges-Louis Baron, nous estimons à propos des technologies en générale et de l'utilisation d'un logiciel de simulation en particulier qu'il y aura modification des tâches et mieux encore des rôles. L'introduction de notre artefact computationnel produira un changement non seulement au niveau des rapports entre l'enseignant et l'apprenant, l'enseignant et le savoir, mais également concernent le savoir et l'apprenant. La saisie des pseudo-codes dans l'interface d'une application de simulation d'algorithmes est une tâche que ni l'enseignant, ni l'apprenant n'effectuait au cours de la rédaction d'un algorithme sur un support en papier. L'enseignant sans être trop directif doit laisser l'apprenant découvrir, explorer pour finalement apprendre de lui-même en le guidant. Ces changements susciteront indéniablement un nouveau questionnement sur le processus enseignement-apprentissage de l'algorithmique. Toutes choses allant indubitablement dans le sens de l'embellissement de la qualité de travail pour l'enseignant et de l'apprentissage pour l'apprenant.

Cependant, il est de tradition dans les sciences de l'éducation d'inscrire tout nouveau dispositif d'apprentissage dans les théories dites de l'apprentissage, soit celles déjà connues, soit d'en faire naitre une nouvelle.

# 1.6. Théories de l'apprentissage

Les théories de l'apprentissage se fixent l'ambition d'expliquer ce qui se passe lors du processus d'apprentissage. Les théories de l'apprentissage sont indispensables pour deux principales raisons : elles offrent un cadre conceptuel permettant d'interpréter des observations et elles permettent des orientations afin de trouver des solutions aux problèmes rencontrés. Au fil du temps, les théories de l'apprentissage ont connu de nombreuses évolutions, ces évolutions portent à la fois sur les finalités de l'apprentissage, le rôle de l'apprenant et de l'enseignant et celui du processus cognitif interne au cerveau (Villiot-Leclercq, 2007).

Quelles sont donc ces différentes théories de l'apprentissage et quelles évolutions ont elles connues ?

## 1.6.1. Béhaviorisme, cognitivisme, constructivisme, socioconstructivisme et connectivisme

Le béhaviorisme encore désigné par le comportementalisme est une théorie de l'apprentissage dont le champ d'intérêt est l'étude des comportements observables. Il n'interroge pas des mécanismes internes au cerveau ou à des processus mentaux qui ne sont pas directement observables (Good et Brophy, 1995). John Watson est toujours considéré comme le précurseur du béhaviorisme, il considérait que faire des expériences de laboratoires en psychologie était la seule procédure objective pour faire de la psychologie en général une discipline scientifique en vue d'établir des résultats exploitables statistiquement (Watson, 1972). Le behaviorisme cadre parfaitement avec une introduction de l'ordinateur dans le processus enseignement-apprentissage avec des applications comme les exerciseurs, les quizz, les jeux éducatifs, etc.

Le cognitivisme encore désigné par le vocable rationalisme naît en 1956 en même temps que l'intelligence artificielle. Il est proposé en réaction au béhaviorisme par Miller et Bruner. Il centre sa démarche sur les manières de penser (ou de « cogiter ») et de résoudre des problèmes. Dans la pensée cognitiviste, l'accent est mis sur l'étude du fonctionnement de l'intelligence, de l'origine de nos connaissances ainsi que des stratégies employées pour assimiler, retenir et réinvestir les connaissances (Legendre, 1993 ; Gauthier et Tardif, 2000). Le cognitivisme s'inspire du modèle de fonctionnement de l'ordinateur pour expliquer la façon dont la mémoire recueille, traite et emmagasine les nouvelles informations et repère par la suite, ces informations (Goupil et Lusignan, 1993). Pour cette théorie, l'enseignant guide, anime, dirige, conseille, explique, régule et remédie, c'est un gestionnaire des apprentissages. Le cognitivisme préconise que l'apprenant intègre à ses schémas mentaux des connaissances qui deviennent une réalité externe pour les réutiliser (Bibeau, 2007). La motivation des élèves pour effectuer les apprentissages dans le modèle cognitiviste est un facteur déterminant. Le cognitivisme suppose la construction de liens entre les anciennes informations déjà organisées en mémoire et les nouvelles données, ce qui exige que les connaissances

soient organisées de façon permanente et sous-entend que soit mobilisées des stratégies cognitives et métacognitives relatives aux savoirs disciplinaires à apprendre.

Finalement, les connaissances procédurales, déclaratives et conditionnelles sont issues de cet l'apprentissage (Tardif, 1992). Pour l'enseignement, il est maintenant question d'instaurer un environnement didactique respectant ces principes de base, qui prend en compte les connaissances antérieures de l'élève, ledit environnement est axé sur l'organisation des connaissances et la mise en place de situation d'apprentissage suscitant l'exécution des tâches complexes, de résolution de problèmes, de transfert, etc. Concernant le rôle de l'enseignant, ce dernier devient concepteur, gestionnaire, entraîneur, médiateur et motivateur. De plus, dans la perception cognitiviste de l'enseignement, il faut multiplier les cheminements d'apprentissage car chaque apprenant est unique et différent donc traite l'information d'une manière personnelle.

Les constructivistes estiment que chaque apprenant interprète la réalité en s'appuyant sur ce qu'il a compris lors de ses expériences passées. Selon le modèle constructiviste, l'acquisition de connaissance ne se réalise pas par simple empilement mais passe par une réorganisation des premières conceptions mentales qui sont construites ou reconstruites. Pour Doolittle (1999), il existe huit conditions importantes pour réussir une pédagogie constructiviste : Mettre les élèves face à des situations d'apprentissage ardues proches de celles rencontrées dans la vie courante. Il faut encourager les actions communes et la collaboration entre les apprenants. Les apprentissages des élèves doivent avoir une signification et les connaissances des élèves doivent être à l'origine de tout apprentissage. Les élèves doivent disposer d'une évaluation formative continue et doivent être comptables de leurs apprentissages. Quant aux enseignants, ils doivent faciliter l'apprentissage en tant que guides. Les contenus revus et diversifiés doivent être présentés. Pour Da Costa (2014), le constructivisme demeure prometteur du point de vue des technologies éducatives car dans sa pratique, il donne une grande autonomie à l'élève et lui permet d'avancer à son rythme en utilisant des outils collaboratifs ou coopératifs. Ce modèle favorise aussi le développement des problèmes assistés par ordinateur.

L'apprentissage socioconstructiviste est perçu comme l'obtention de connaissances à l'aide des échanges entre l'enseignant et les élèves ou entre élèves. Les apprenants n'acquièrent pas des connaissances uniquement grâce à leur transmission par l'enseignant mais aussi grâce aux interactions. Selon ce modèle, pour se développer, les apprentissages doivent être circonscrits dans une zone. Le rôle du maître revient donc à définir précisément cette zone afin de donner des exercices appropriés. De plus, il va favoriser le débat entre les élèves, générant ainsi des « conflits sociocognitifs », en les faisant travailler en groupe. L'approche socioconstructiviste estime aussi que les erreurs correspondent à un point d'appui pour la construction de nouvelles connaissances. Bruner (1996) a expliqué que dans le modèle transmissif l'enseignant monopolise le savoir, ce qui empêche l'acquisition de l'autonomie des élèves. Pour lui, l'enseignant doit rendre la tâche plus agréable à réaliser avec son aide tout en évitant que l'élève

devienne dépendant de lui. Il doit aussi captiver l'attention de l'apprenant en maintenant l'intérêt de la tâche.

Pour Siemens (2005), le connectivisme se décrit comme l'ensemble des principes issus de la théorie du chaos, des réseaux, de l'auto-organisation et de la complexité. Dans un tel contexte, l'apprentissage est un processus se produisant dans des environnements flous composés d'éléments de supports variés, et n'étant pas totalement contrôlé par l'individu. L'apprentissage peut échapper au contrôle de l'individu (au sein d'une organisation ou une base de données), et met l'accent sur les liens entre des informations spécialisées. Ces liens sont plus importants que l'état actuel de notre connaissance car ils favorisent une grande compréhension. Le connectivisme tire son intérêt de ce qui représente également son inconvénient majeur, du fait que les éléments sur lesquels on s'appuie pour prendre des décisions varient rapidement.

## 1.6.2. Théorie de l'apprentissage multimédia

La théorie de l'apprentissage multimédia établie par Mayer (2009) est axée sur les apprentissages et les capacités de l'utilisateur et non pas sur les performances que ce dernier peut avoir grâce aux outils multimédias. La théorie de l'apprentissage multimédia met en exergue l'idée d'apprentissage, et soutient le fait que l'apprenant va apprendre plus facilement s'il a été associé à la conception du support d'apprentissage (Mayer, 2009).

Trois aspects principaux originaires de la psychologie cognitive sous-tendent la théorie de l'apprentissage multimédia. On a d'abord le double codage (Païvio, 1986 ; Baddeley, 1992) qui défend qu'il existe deux voies entre l'auditif et le visuel, puis entre le verbal et le non-verbal. Chaque voie a la possibilité d'échanger avec l'autre. L'utilisation des deux s'avère donc importante. Le second aspect est celui de la capacité limitée de stockage (Baddeley, 1992 ; Chandler et Sweller, 1991). On admet que la somme d'informations stockable à l'intérieur des différents canaux (auditif, visuel, mémoire de travail...) est limitée. Enfin le dernier aspect est celui du processus actif (Mayer, 2008a ; Wittrock, 1989) qui soutient le fait qu'il y a apprentissage uniquement quand l'élève est protagoniste de son apprentissage. C'est ainsi que l'apprenant va choisir l'information importante, l'organiser et l'intégrer avec les connaissances antérieures afin de bâtir un modèle mental complet servant de support de manipulation pour des informations futures. En définitive, la théorie de l'apprentissage multimédia de Mayer (2009) se base sur les trois aspects précédents et correspond au schéma indiqué ci-dessous. Elle nous enseigne qu'il y a deux catégories de messages dans un apprentissage multimédia : des mots et des images. Chaque stimulus va être appréhendé et donc entrer dans la mémoire sensorielle lui correspondant. La théorie du double codage annonce ici la présence de deux mémoires différentes pour les stimuli visuels et auditifs. Chaque élément de mémoire sera ensuite introduit dans la mémoire de travail qui dispose de capacités de traitement limitées et passera soit par la voie supérieure (sons, verbal) pour les mots saisis, soit par la voie inférieure (images, pictural) pour les

images vues, soit alternera entre les deux voies pour le son imagé (ex : musique). Nous pouvons signaler que les mots peuvent être traités par l'intermédiaire de deux canaux différents selon qu'ils sont lus ou entendus. L'introduction simultanée des mots lus et des images vues va surcharger la mémoire sensorielle visuelle et de fait entraver l'apprentissage. C'est sensiblement le même phénomène que nous tentons de comprendre dans le cadre de l'enseignement-apprentissage de l'algorithmique. En effet, la double activité cognitive que représente la résolution du problème et l'apprentissage du formalisme fait entrave à l'apprentissage, par surcharge cognitive.

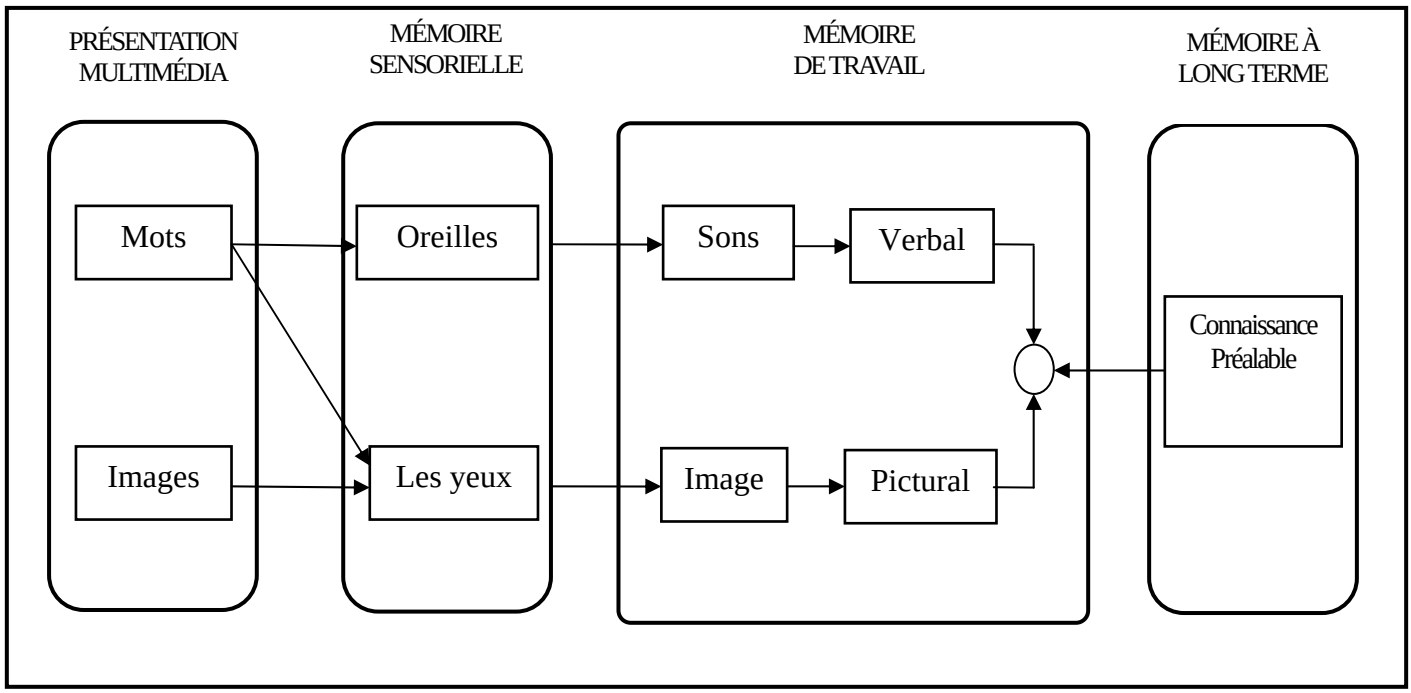



*Figure 1 : La théorie de l'apprentissage multimédia selon Mayer (2009)*

En résumé, le béhaviorisme s'intéresse plus à l'information à transmettre qu'au système cognitif de l'apprenant. Il préconise la définition des objectifs et des progressions prédéterminés, établis à partir d'un découpage strict du savoir. Le cognitivisme, quant à lui, se concentre sur le rôle de la mémoire, l'organisation des connaissances en mémoire et le traitement de l'information. Par conséquent, l'apprentissage qui était l'acquisition de connaissances devient la construction de connaissances. De même, l'enseignement qui consistait à faire de la transmission de connaissances, sert désormais d'aide à la construction des connaissances. Les outils informatiques construits dans cette perspective offrent alors la possibilité d'un enseignement personnalisé, une plus grande interactivité et offrent des activités plus complexes. Enfin, les outils qui sont orientés vers un apprentissage collaboratif, qui implique une interaction verbale entre les apprenants, ainsi que ceux qui favorisent l'interaction apprenant / objet du monde s'appuie sur les théories du constructivisme du socioconstructivisme. Si les théories de l'apprentissage ont impacté la conception des

outils informatiques, les recherches concernant l'apprentissage multimédia ont, quant à elles, ont eu un rôle dans la conception des modèles d'enseignement.

## 1.6.3. Théorie de la charge cognitive

La théorie de la charge cognitive est aussi essentielle comme support théorique de notre travail. En effet dans l'une de nos hypothèses, il est question que notre artefact computationnel pourrait faciliter l'apprentissage de l'algorithmique en soulageant en partie l'apprenant de l'activité liée à l'apprentissage du formalisme par réduction de la charge cognitive.

Ce sont les psychologues John Sweller et Fred Paas qui ont développé la théorie de la charge cognitive. Cette théorie vise à expliquer les échecs ou les réussites des personnes en activité d'apprentissage en situation de résolution de problème. La charge cognitive mesure la quantité de ressources mentales mobilisées par un sujet pour accomplir une tâche. La charge cognitive dépend de l'individu qui doit réaliser la tâche, de la complexité de la tâche et de l'environnement dans lequel le sujet accomplit la tâche. Dans une situation d'apprentissage, l'enseignant doit avoir pour but d'éviter les processus pédagogiques coûteux en charge cognitive. L'architecture cognitive s'organise autour d'une mémoire à long terme (MLT) qui dispose d'une capacité de stockage quasi illimitée et d'une mémoire de travail (MDT) ayant une capacité limitée pour les connaissances nouvelles. Ainsi, le but de l'enseignement est d'augmenter la quantité et la qualité des connaissances utilisables en mémoire à long terme, de cette façon, si rien n'a varié en MLT cela signifie dans ce cas que rien n'a été appris (Sweller, 1988).

La charge cognitive est tributaire de ce qui est présenté (charge intrinsèque) et de la façon dont cela est présenté (charge extrinsèque). La charge intrinsèque et la charge extrinsèque s'ajoutent. La charge intrinsèque d'une tâche peut être acceptable en mémoire de travail, mais si elle s'additionne à une grande charge extrinsèque, le sujet sera en surcharge cognitive (Sweller, 1988).

La charge intrinsèque est attachée à la tâche en elle-même, et ne peut être soulagée qu'en adaptant la tâche aux connaissances de l'apprenant. Cependant Schnotz (2008) note qu'une même tâche accomplie par des novices ou des experts n'engendrera pas la même surcharge cognitive : les experts possèdent plusieurs schémas mentaux, qui peuvent associer beaucoup d'informations dans un seul élément, considéré comme un ensemble cohérent par la mémoire de travail. On établirait de cette manière que l'expertise permet de diminuer la charge intrinsèque.

La charge extrinsèque est modifiable car elle est liée à la manière dont l'information est présentée. Elle peut être diminuée par une révision du matériel à apprendre. La suppression d'éléments inutiles ou superflus permet de réduire la charge extrinsèque, sans toucher à la charge intrinsèque. Une tâche qui oblige au traitement quasi simultané d'informations éloignées les unes des autres pour être accomplie augmente la charge

cognitive (Schnotz, 2008). Une présentation au même moment d'informations permettrait donc de réduire la charge cognitive.

Il existe une dernière catégorie de charge cognitive : la charge essentielle. La charge cognitive essentielle sert à inclure les connaissances en mémoire à long terme, sous forme de schémas mentaux. Aussi, il y a apprentissage, si la charge intrinsèque et la charge extrinsèque sont allégées, en d'autres termes, toute activité d'apprentissage implique une charge cognitive, c'est la charge essentielle.

A quoi sert donc la théorie de la charge cognitive dans la compréhension du fonctionnement du processus enseignement-apprentissage ?

Dans le processus enseignement-apprentissage, la théorie de la charge cognitive permet plusieurs implications d'ordre pédagogiques, parmi ces implications, l'une d'entres elles est le fait qu'elle facilite l'apprentissage. De ce fait, il est possible d'opter pour une modification de la charge cognitive extrinsèque afin de faciliter l'apprentissage, et proposer des aménagements de la tâche pour la diminution de la charge cognitive intrinsèque.

Par exemple, travailler à partir d'exemples s'avère moins coûteux en termes de charge cognitive que de devoir résoudre un problème seul. Toutefois, lorsque les apprenants disposent d'une expertise dans le domaine étudié, cette orientation peut s'inverser. On observe dans ce cas une redondance entre les exemples en mémoire et ceux qui sont actuels. On peut aussi traduire cette répétition non comme une surcharge, mais comme une gêne pour construire un nouveau schéma mental : il n'y a pas de motivation à apprendre puisque la tâche est simple. Répéter des exemples travaillés n'est pas dans cette perspective très utile, il est alors préférable de laisser l'élève automatiser ses schémas avec de nombreux exercices et situations de pratique autonome. En conséquence, les exemples travaillés doivent être suivis d'exercices autonomes. Plus un apprenant a de connaissances préalables, moins les exemples travaillés lui seront utiles (Sweller, 1988).

Les effets de la théorie de la charge cognitive se trouvent au centre de notre réflexion sur le processus enseignement-apprentissage de l'algorithmique. En effet, l'enseignement de l'algorithmique et/ou d'un langage informatique s'appuie sur des problèmes à résoudre. L'enseignement-apprentissage de l'algorithmique et des langages informatiques impliquent alors chez l'apprenant deux activités cognitives : l'apprentissage du formalisme de l'algorithmique, puis d'un langage informatique, ainsi qu'une démarche de résolution de problème. Dans cette situation, l'apprenant doit, dans un premier temps, apprendre le formalisme de l'algorithmique, puis celui du langage informatique, et après, à solutionner le problème posé par la tâche prescrite. Cette double activité cognitive ne simplifie pas chez les élèves l'apprentissage en raison de la charge cognitive générée par ces deux activités (Amadieu & Tricot, 2006). Il convient donc de soulager l'apprenant de cette charge cognitive élevée afin de faciliter l'apprentissage. Les travaux de John Sweller ont ainsi établi que certaines approches

dans l'enseignement-apprentissage n'ont pas un coût cognitif très élevé, de ce fait, elles facilitent aussi l'apprentissage. C'est ainsi qu'il a été démontré l'efficacité didactique des problèmes sans but spécifié par rapport à ceux ayant un but. Lorsqu'on donne clairement le but du problème à un sujet qui ne possède pas le schéma pour résoudre ce problème, immédiatement on induit chez ce dernier une démarche d'analyse qui peut être extrêmement coûteuse cognitivement et aboutir à l'absence d'apprentissage. L'utilisation de deux sens différents facilite également l'apprentissage dans la mesure où la charge de la mémoire de travail est moins importante lorsque le même matériel est présenté en utilisant la vue et l'ouïe, que quand le seul canal visuel est utilisé. Une extension de cette approche est aussi utilisée dans l'usage d'un simulateur d'algorithme car voir s'exécuter à l'écran un algorithme que l'on a écrit sur une feuille augmente certainement par ce canal visuel la certitude de la réussite de l'apprentissage. Le travail sur des exemples de problèmes résolus entraine également un meilleur apprentissage car il y a baisse de charge cognitive dans la mesure où cette approche entraine une focalisation de l'attention sur les états du problème et sur les opérateurs associés. Dans l'apprentissage en l'algorithmique, on utilise souvent des exemples de problèmes résolus. A l'entame du cours on présente souvent un exemple de problème résolus pour montrer la structure générale d'un algorithme, des exemples résolus servent aussi à introduire de nouvelles structures (répétition ou itération) et règles syntaxiques. Avant l'utilisation d'un logiciel de simulation d'algorithme (AlgoBox par exemple), l'apprenant doit avoir un exemple résolu de l'algorithme qu'il souhaite simuler. L'autre approche consiste à introduire des informations sous forme de texte et d'images dans le problème posé afin d'éliminer l'effet de dissociation de l'attention, les effets conjugués des exemples de problèmes résolus et de l'intégration d'informations facilitent l'apprentissage. On peut donc déduire que travailler avec les apprenants sur des exemples de problèmes résolus, par l'effet de la diminution de la charge cognitive facilite la résolution de problèmes. La résolution de problèmes étant en algorithmique et en programmation l'exercice qui permet de déterminer si l'apprenant a appris.

Quand il parle de résoudre un problème, Richard (2005) évoque le fait qu'il s'agit de produire les raisonnements nécessaires afin de pouvoir identifier les actions possibles et les ordonner. Ainsi, il faut donc avoir un ensemble de connaissances qui soient liées à la situation problème afin de produire le raisonnement dont on a besoin pour pouvoir résoudre le problème posé par la tâche prescrite. Par conséquent, la résolution de problème est une activité mentale conduisant à la compréhension d'une situation qui pose problème au sujet et aboutissant à une action. Autrement dit, il s'agit de la mise en œuvre de procédures connues, sous la forme d'une construction nouvelle car intégrant les éléments de la situation (Mayer, 2008), et produite par raisonnement.

La présente réflexion autour du processus enseignement-apprentissage de l'algorithmique voire des langages informatiques en général conduit donc à s'interroger sur ces mécanismes du raisonnement et de compréhension qui aboutissent *in fine* à l'élaboration de la solution au problème. De ce fait, l'apprentissage de l'algorithmique, voire des langages informatiques, suggère deux activités qui peuvent être concurrentes :

l'apprentissage des règles syntaxiques et le mécanisme d'analyse du problème à résoudre, ces deux activités ont la possibilité de mettre les apprenants en situation d'échec. Ce, d'autant, que Sweller (1988) a établi que les processus cognitifs qui sont impliqués dans la résolution d'un problème se distinguent de ceux qui sont impliqués dans l'acquisition de connaissances. En effet, ici il est question d'acquisition de connaissances procédurales, les procédures connues sont adaptées aux nouvelles situations et de nouvelles procédures sont élaborées en faisant des essais exploratoires, en envisageant des hypothèses du fonctionnement du système, en vérifiant la validité des actions avant d'intégrer hiérarchiquement les nouvelles procédures. En revanche, résoudre un problème ce n'est pas seulement élaborer une procédure, c'est trouver le codage de chacun des objets de la situation et l'interprétation d'ensemble qui est compatible avec les contraintes du problème. D'une manière imagée, nous dirons que l'acquisition de connaissances ici consiste à élaborer des fiches et leur contenu, tout en adaptant le contenu d'autres fiches, alors que résoudre un problème c'est aller choisir dans une ou plusieurs fiches les contenus qui cadrent avec la situation afin de les traduire en fonction du contexte. L'élaboration de ce processus de résolution aux problèmes posés peut donc être coûteux cognitivement pour un élève en situation d'apprentissage (Tricot, 2003). En effet, un problème peut être facile pour un apprenant qui dispose des connaissances appropriées et très difficile pour un autre ne disposant pas de ces connaissances, ou si le contexte spécifique de la situation-problème ne ressemble à rien de connu pour l'apprenant (Cauzimille-Marmèche, 1991 ; 1996).

En résumé, l'apprentissage de l'élaboration d'un algorithme impose au sujet une charge cognitive qui correspond au maintien en mémoire des informations concernant le problème posé et celles ayant un rapport avec l'écriture du formalisme de l'algorithme. Si le sujet est novice dans l'élaboration des algorithmes et/ou le sujet ne dispose pas des connaissances relatives au problème posé à cet instant, cela rend donc l'apprentissage difficile. Pour surmonter ces obstacles, l'utilisation d'un intermédiaire graphique dans le processus enseignement-apprentissage de l'algorithmique permet de projeter une diminution de la charge cognitive et par conséquent améliorer les apprentissages des élèves.

Mayer (2001) a mis en évidence trois phases de traitement de l'information pour l'apprentissage multimédia : il y a la phase de sélection dans laquelle les images, figures et animations sont prises en charge par la mémoire iconique (partie visuelle). Les mots peuvent quant à eux être traités soit par la mémoire échoïque (auditive) s'il s'agit d'un commentaire, soit par la mémoire iconique s'il s'agit d'un texte écrit. La procédure de sélection correspond au relais vers la mémoire de travail des informations jugées pertinentes pour la tâche. Dans la phase d'organisation, chaque élément d'information perçu et identifié, doit être lié aux autres. Les relations peuvent être causales, temporelles, logiques, etc. Reconnaître une absence de relation s'avère également primordial. Quant à la phase d'intégration, les informations picturales et verbales qui ont été organisées séparément sont intégrées et liées à des connaissances antérieures pour ne former qu'un seul modèle mental. La mémoire à long terme intervient pour

fournir les connaissances antérieures, l'intégration se fait en mémoire de travail et il en résulte un nouveau modèle mental, élargi et indicé qui est alors stocké en mémoire à long terme. On peut alors considérer que les informations ont été apprises.

Cependant, Mayer (2003) insiste sur le fait que les trois phases décrites ne surviennent pas dans un ordre linéaire rigide. Il s'agit plutôt d'un processus itératif, fait d'aller et venue entre sélection, organisation et intégration. Ceci d'autant plus que les capacités limitées de la mémoire de travail ne permettent pas de prendre en compte la totalité des informations pour les organiser en une seule fois. Mayer démontre ainsi que dans un processus enseignement-apprentissage multimédia, l'acquisition des connaissances est facilitée du fait de la variété de types de présentation de l'information (image, figures, animations, etc.), ce qui a pour conséquence de solliciter des mémoires différentes (mémoire visuelle, mémoire auditive) évitant donc d'élever la charge cognitive.

En s'appuyant sur ses travaux, Mayer (2001) a aussi dégagé certains principes dont il faut tenir compte pour élaborer et présenter des informations en vue d'un apprentissage efficace et facile. Il y a le principe multimédia qui stipule qu'une information constituée de mots et d'images appropriées est mieux apprise que celle ne disposant que de mots. Le second principe est celui de la contiguïté spatiale qui dit que l'apprentissage est plus efficace lorsque les images et les mots correspondants sont présentés de manière rapprochée. La contiguïté temporelle conditionne la performance de l'apprentissage à la présentation simultanée des éléments verbaux et visuels. Le principe de cohérence voudrait que les mots, images et sons non utiles ne soient pas présents. La modalité propose que soient présentées les informations accompagnées d'un commentaire audio afin obtenir de meilleurs résultats. Dans le principe de la redondance, l'apprentissage est meilleur lorsqu'un commentaire audio accompagne une information plutôt qu'une information accompagnée d'un texte à l'écran. Enfin la différence individuelle dit que les effets sont plus forts sur les utilisateurs novices. Tous ces principes cadres parfaitement avec l'utilisation dans le processus enseignement-apprentissage de l'algorithmique avec le logiciel d'aide à l'élaboration d'algorithme « AlgoBox » à l'exception de la modalité, la redondance car cette application ne dispose pas de dispositif audio pour permettre d'accompagner les informations avec des commentaires adéquats. Cependant, l'enseignant peut compenser cela lors de la simulation en donnant des explications.

Au terme de ce sous-chapitre consacré aux théories de l'apprentissage, nous retiendrons que ces théories visent comme objectif d'expliquer ce qui se passe lors du processus d'apprentissage. Que l'on se situe du point de vu des théories plus anciennes ou de celui des plus ressentes comme celle de l'apprentissage multimédia ou de la charge cognitive, expliquer le processus d'apprentissage conduira à s'intéresser aux types de savoirs qui seront mise en œuvre dans ce processus d'apprentissage. Dans le cadre de notre recherche, quelles sont alors les savoirs mis en œuvre pour l'apprentissage de l'algorithmique ? Dans l'accomplissement d'une tâche liée à l'écriture d'un algorithme, l'apprenant fait appel à deux types de savoirs portés par la tâche

prescrite. Ceux qui peuvent être présents dans la tâche mais que l'enseignant n'a pas forcément pris en compte car pour lui il est évident que l'élève les maitrise : ce sont des « savoirs implicites ». En revanche, ceux que l'enseignant souhaite enseigner, c'est-à-dire qu'ils sont visés par la tâche et qui sont directement enseignés au cours d'algorithmique sont dits « savoirs explicites ».

## 1.7. Connaissances implicites et connaissances explicites

Dans l'enseignement-apprentissage, on fait largement échos des notions d' «implicite» et d' « explicite ». En psychologie, l'implicite renvoie à une dimension de soi difficilement accrochable, à peine évidente et pourtant manifeste. A ce niveau, les enjeux ne sont pas encore explicités (conscientisés, reconnus). La dimension explicite déploie des façons de pensées ou autres modes (métaphorique, gestuel, artistique, etc.) un sens contenu implicitement (Gendlin, 2010).

Dans l'accomplissement d'une tâche, ce qui est implicite pour l'apprenant renvoie à son expérience vécue globalement en deçà de tout ce qu'on peut en dire. Cette expérience est néanmoins porteuse de sens et de signification pour l'apprenant. Ce sens demande à être explicité afin qu'il serve dans l'accomplissement d'une tâche. L'explicitation s'appuie sur un processus d'évaluation interne. En effet, en se servant de toutes les expériences vécues et sur lesquelles il peut s'appuyer, l'apprenant trouve, dans la dimension implicite, une base à laquelle il peut revenir pour former de nouvelles idées, découvrir de nouvelles orientations (Gendlin, 2010). Il va trouver, à partir de là, comment modifier et ajuster ses comportements, trouver une solution à un problème.

Les expériences vécues par l'apprenant peuvent être des apprentissages qui ont été soit formels, c'est-à-dire dans le cadre scolaire, soit informels c'est-à-dire dans un cadre ou l'intention n'était pas de susciter un apprentissage chez l'apprenant. Cependant, formel ou informel s'il y a eu apprentissage, l'apprenant sera capable d'exécuter la tâche en référence à l'apprentissage. La réalisation d'un tâche est perçue comme une action du fait de l'activité du sujet, c'est le résultat de l'analyse et de l'entendement du langage qui appelle une réponse en acte (Richards, Platt & Weber, 1985), attestant que le sens a été perçu pour atteindre l'objectif visé (Bygate, Skehan & Swain, 2001). La tâche réalisée est la matérialisation de l'activité, elle est accomplie par les apprenants pour avoir une œuvre palpable à partir du traitement cognitif d'informations, cette réalisation permet alors à l'enseignant de vérifier et réguler ce processus (Prabhu, 1987).

D'une manière générale, les connaissances implicites sont des connaissances dont la personne (l'apprenant) n'a pas conscience, elles ne peuvent être verbalisées et elles suscitent un important sentiment d'intuition, puisque l'apprenant n'est pas conscient de son savoir alors même qu'il démontre la capacité à l'utiliser. Les connaissances implicites s'avèrent également durables car elles sont peu touchées par le temps ou par une tâche secondaire (Gasparini, 2004). Des recherches récentes démontrent que

l'apprenant acquiert des connaissances implicites grâce à la fréquence d'association de certains éléments, et ce, indépendamment de l'intention de l'apprenant (Perruchet & Pacton, 2004 ; Gombert, 2006 ; Lété, 2006 ; Deacon, Conrad & Pacton, 2008). De plus selon Perruchet & Nicolas (1998), être en contact de façon régulière avec uniquement des exemples positifs facilite l'acquisition des connaissances implicites. Cette situation entraine également d'importantes conséquences pour l'enseignement, précisément dans le traitement des erreurs. Ici il ne faudrait pas attirer l'attention de l'apprenant sur l'erreur (Rey, Pacton & Perruchet, 2005). En effet, si l'intention d'apprendre n'influence pas la qualité de l'acquisition des connaissances implicites, l'attention y joue un rôle essentiel (Perruchet & Pacton, 2004). En définitive, pour assoir les connaissances implicites, l'enseignant doit exposer les apprenants à de nombreux exemples des structures à mémoriser en attirant leur attention sur ces structures sans qu'ils aient nécessairement l'intention d'apprendre et surtout sans exposer les élèves aux erreurs mais seulement à des exemples positifs.

Concernant les connaissances explicites, généralement ce sont celles que l'apprenant est en mesure de verbaliser et de contrôler intentionnellement. Les apprentissages explicites se caractérisent par le fait d'être conscient qu'on apprend. Le fait donc de savoir qu'on apprend valorise la performance réalisée dans une tâche accomplie en s'appuyant sur des connaissances explicites. Les connaissances explicites ne concernent pas uniquement les connaissances déclaratives, mais aussi des procédures que le sujet applique consciemment. Les connaissances explicites sont coûteuses sur le plan cognitif en début d'apprentissage, mais heureusement à la longue, elles s'utilisent de plus en plus facilement et exigent moins d'attention (Gombert, 2006). Le mode d'apprentissage des connaissances explicites diffère grandement de celui des connaissances implicites. Il présuppose un mode de pensée hypothético-déductif tel qu'adopté en résolution de problèmes et implique la formulation d'hypothèses et leur vérification, ainsi que l'application de règles explicites et d'autres processus conscients similaires (Gasparini, 2004). Il s'agit d'un mode d'apprentissage efficace si les éléments à mettre en relation peuvent être assez facilement discriminés, si l'information pertinente ne se trouve pas brouillée parmi un trop grand nombre de facteurs qui dépasserait alors les limites de ce qu'on peur traiter consciemment (Gasparini, 2004).

Dans l'acquisition des connaissances explicites, le statut de l'erreur se transforme, l'erreur devient le miroir des représentations des apprenants. En comprenant leurs erreurs, il devient possible de confronter et de faire évoluer les conceptions des apprenants (Reuter, 1996). Finalement, l'apprentissage des concepts abstraits passe de l'observation d'exemples à l'abstraction de connaissances au moyen d'une réflexion guidée par les échanges verbaux avec une personne expérimentée pour pouvoir ensuite être transférées dans de nouveaux contextes.

En résumé, les connaissances implicites et les connaissances explicites conduisent quelque fois à des interventions opposées, particulièrement dans la façon de traiter l'erreur. Les connaissances explicites sont essentielles, mais les connaissances implicites

rendent également service en raison du peu d'attention qu'elles requièrent une fois acquises, mais également à cause du fait que certaines règles trop complexes à détailler explicitement réussissent à être apprises implicitement. Malheureusement, cette situation ne se rencontre pas avec toutes les règles compliquées.

Les connaissances explicites ne se limitent pas uniquement à des connaissances déclaratives des règles, mais touchent également la connaissance explicite des procédures et des manipulations qui offrent la possibilité de résoudre des problèmes.

Dans notre expérimentation, la résolution des problèmes d'algorithmique va imposer chez les sujets l'utilisation des connaissances émanant de savoirs enseignées ou pas en cours. Ce sont ces connaissances qui vont constituer les indicateurs de réussite ou de non-réussite. Que la connaissance soit implicite ou explicite, nous ne nous intéresserons uniquement qu'au fait que le sujet ait été capable ou pas de la mettre en œuvre. Puis, conformément au fait que nous essayons de comprendre les difficultés d'un processus enseignement-apprentissage, les cas d'échecs seront aussi examinés par rapport à la nature de la connaissance (implicite ou explicite). En d'autres termes, s'interroger sur le type de connaissances qui suscitent le plus d'échec. Cependant, la nature des problèmes en algorithmique nous oblige aussi à établir le fait que la double activité cognitive (formalisme et résolution de problème) que cette tâche suscite s'appuie sur des connaissances aussi bien implicites qu'explicites. Les connaissances explicites sont surtout sollicités dans le respect du formalisme de l'algorithme car ces règles on été apprises de manières explicite ou à l'aide d'exemples au cours d'algorithmique et aussi en travaux dirigés. Mais la résolution de problèmes proprement dite peut faire appel à des connaissances implicites, pas apprises en algorithmique mais dans d'autre disciplines. En d'autres termes, les connaissances implicites sont les savoirs portés par la tâche prescrite mais qui ne correspondent pas aux connaissances ou savoirs que l'enseignant veut tester. Pour illustrer notre propos, lorsqu'au cours un problème de géométrie qui consiste pour l'élève à tracer un triangle, le savoir implicite, ici, qui est d'ailleurs un savoir-faire, c'est que le tracé doit se faire impérativement au compas et non à la règle.

Au terme de ce sous-chapitre sur les connaissances implicites et explicites, qui vient aussi clôturer toutes les réflexions faites sur le processus cognitif mis en œuvre dans l'enseignement-apprentissage de l'algorithmique, il nous plait de rappeler que notre réflexion sur ce processus doit aussi porter sur l'artefact utilisé dans l'enseignement-apprentissage de l'algorithmique afin de le faciliter. C'est donc dans cette optique que nous présentons les différentes utilisations de l'ordinateur dans l'enseignement ainsi que les liens entre ces usages et certains concepts qui sont au centre de notre recherche comme la résolution de problèmes et la charge cognitive.

# 2. Outils numériques dans l'apprentissage de l'algorithmique

## 2.1. Usages de l'ordinateur dans l'enseignement

L'essor de l'informatique pédagogique repose sur diverses origines de contraintes matérielles, institutionnelles et humaines. L'enseignant qui opte pour l'intégration de l'ordinateur dans sa classe n'a initialement qu'une influence limitée sur ces contraintes. Il est contraint de suivre l'évolution des matériels et des langages, il obéit aux décisions politiques en matière de ressource informatique, plus qu'il ne participe à celle-ci et il assiste aux mutations humaines, plus qu'il ne les détermine.

Il nous semble important de se poser d'emblée la question de savoir quels sont alors les problèmes d'enseignement-apprentissage rencontrés par les enseignants dans la mesure où les prescriptions institutionnelles demandent aux enseignants d'utiliser les outils informatiques, voire pour certains enseignants d'enseigner l'informatique.

### 2.1.1. Informatique pédagogique, simulation informatique et l'école

L'informatique pédagogique, c'est le croisement d'une réalité psychologique, c'est à dire un sujet en situation d'apprentissage, et d'une réalité institutionnelle dans le cadre de l'école par exemple, avec un outil technologique, c'est à dire l'ordinateur et ses logiciels par le biais d'un contenu qu'il faut communiquer (les savoirs à enseigner).

Le développement de techniques informatiques avancées en simulation influence indubitablement le développement des logiciels d'enseignement et des environnements d'apprentissage. Nous l'avons notamment constaté ces dernières années dans les langages et les outils de développement permettant de construire des interfaces graphiques attrayant tout en réduisant quelque peu le travail de programmation.

L'organisation des savoirs en disciplines et la formation des enseignants sont des facteurs auxquels l'institution scolaire doit s'attaquer avant d'intégrer harmonieusement à son programme des activités s'appuyant sur l'ordinateur et, comme objet à enseigner, l'apprentissage d'un nouveau langage.

Les premières applications de l'informatique en éducation ont suscité un débat, souvent vif, entre les défenseurs de l'enseignement programmé ou de Skinner qui veut qu'on apprenne en observant ses propres actes et leurs conséquences, et les tenants d'un apprentissage ouvert sur la découverte. Pour les premiers, l'informatique est un outil efficace exclusivement dans le cadre d'un entraînement et d'une répétition de

séquences d'enseignement. Alors que les seconds estiment que l'informatique est plus un « médium », un support pour construire des environnements dans lesquels l'apprenant bâtit son propre savoir (Papert, 1980). Cette seconde approche semble redevenir d'actualité en France si on se réfère au rapport du programme international pour le suivi des acquis des élèves (PISA) de 2015 : « La technologie peut ainsi renforcer l'apprentissage par l'expérience, favoriser les méthodes pédagogiques d'apprentissage par projet et par investigation, faciliter les activités pratiques et l'apprentissage collaboratif, permettre une évaluation formative en temps réel et soutenir les communautés d'apprentissage et d'enseignement, en offrant de nouveaux outils tels que les laboratoires virtuels et à distance, les didacticiels non linéaires très interactifs fondés sur une conception pédagogique de pointe, les logiciels sophistiqués d'expérimentation et de simulation, les médias sociaux et les jeux sérieux ».

### 2.1.2. Utilisation pédagogique de l'ordinateur, résolution de problème et charge cognitive

L'enseignement-apprentissage de l'algorithmique en se servant d'une application d'aide à l'élaboration d'algorithme est une utilisation pédagogique de l'ordinateur, cette utilisation comme beaucoup d'autre favorise les méthodes pédagogiques d'apprentissage par projet et surtout la résolution de problèmes. Les études sur la résolution de problèmes ont révélé plusieurs étapes dans le processus complexe qui conduit un sujet de l'énoncé d'un problème à sa solution. La première de ces étapes est identifiée comme la phase de « représentation du problème ». Dans la représentation du problème, l'apprenant va activer un certain nombre de connaissances contenues en mémoire à long terme ; c'est l'interprétation faite par l'apprenant des éléments caractéristiques de la situation. Cette représentation va évoluer par la découverte de sous-buts qui deviendront autant de buts intermédiaires et par la mise en place d'actions qui correspondent à chaque but intermédiaire. Le regroupement des actions de tous les buts intermédiaires sera l'action liée au problème posé. Cette description quelque peu simplifiée de la représentation d'un problème nous permet un tant soit peu de décrire ce qui se passe dans des situations aussi complexes que la programmation informatique. En effet, entre l'énoncé du problème qui est posé et la solution qui est décrite soit à travers un algorithme ou un langage de programmation, la barrière du formalisme ou la structure modulaire (bloc par bloc) ne facilite pas l'établissement d'une passerelle entre les deux. En conséquence, l'apprenant comprend difficilement aussi bien le problème que la solution. Il nous parait donc important de nous interroger sur ce qui, dans le processus enseignement-apprentissage, rend cet apprentissage complexe, le processus cognitif mis en œuvre par l'apprenant pour apprendre, la méthode pédagogique utilisée ou éventuellement les deux. Si la résolution de problèmes est préconisée dans les utilisations pédagogiques de l'ordinateur, que dire alors du processus cognitif mis en œuvre en algorithmique et en programmation ?

Le concept de charge cognitive est associé au problème des contraintes mnésiques imposées par la résolution d'un problème. Tout apprentissage d'un dispositif complexe,

à l'instar de l'algorithmique présente, au départ, des difficultés assez importantes lorsque l'on souhaite en maîtriser toutes les possibilités. En effet, la majeure partie des capacités attentionnelles du sujet est focalisée sur un aspect de la situation problème, délaissant les autres. Finalement, l'attention est portée sur plusieurs traitements élémentaires et il n'est plus possible à l'élève de se centrer sur la principale tâche. C'est l'automatisation de ces traitements élémentaires qui permettra, après un long entraînement, de détourner cette attention sur ce qui est essentiel à la résolution : la gestion des sous-buts et des actions qui en découlent. Ce concept de charge cognitive prend donc ici toute son importance, car il suppose que l'on introduise l'idée d'une certaine hiérarchie de buts dans la manipulation d'un dispositif complexe.

### 2.1.3. Typologie de l'utilisation pédagogique de l'ordinateur

Dans l'acte pédagogique, Il s'établit une relation privilégiée entre l'enseignant et l'élève, c'est cette relation privilégiée qui détermine la réussite de l'apprentissage. Il est nécessaire de détecter pour chaque apprenant les éléments qui facilitent son apprentissage et éventuellement, ce qui peut y faire obstacle. Il paraît donc évident que, quels que soient les progrès encore réalisables, l'informatique ne sera jamais en mesure de capter et de traiter toutes ces informations, et même si cela s'avère possible de tels dispositifs seraient extrêmement lourd et coûteux, de se substituer au professeur car celui-ci a d'autres rôles.

Par contre, l'ordinateur devient un assistant précieux pour aider l'enseignant à alterner ses méthodes. C'est une ressource supplémentaire. L'enseignant choisit l'outil informatique quand il lui apporte quelque chose de nouveau et de mieux par rapport aux autres outils d'enseignement. C'est fort de cette évidence que Bénézra, Jean et Rothan (1989) ont proposé une typologie de six utilisations pédagogiques de l'ordinateur.

Le but de cette typologie est de montrer la souplesse et les multiples utilisations pédagogiques de l'informatique. Il ne faut pas la prendre comme une taxonomie rigoureuse, mais plutôt comme un support servant à construire sa propre utilisation pertinente, efficace, qui nous satisfait au cours de l'élaboration d'une séquence d'apprentissage.

- **Encyclopédie active** : L'utilisation pédagogique de l'ordinateur comme encyclopédie repose sur le fait que l'ordinateur contient un répertoire de nombreux exemples dont se servira l'apprenant et qui lui permettra d'appréhender un phénomène ou un concept. Le but pédagogique de cette utilisation de l'ordinateur est d'approcher un concept par la manipulation de données telles que les images, les graphiques, les textes, les sons, etc. Des logiciels de banques de données avec automatisation de certains traitement à la demande de l'utilisateur ainsi que des applications de simulation de phénomènes cadrent parfaitement avec cet usage (Bénézra, Jean & Rothan, 1989).

- **Affiche évolutive** : Le principe de l'utilisation pédagogique de l'ordinateur en tant qu'affiche évolutive consiste à faire en sorte que l'informatique apporte une aide efficace pour la circulation de l'information entre les groupes. Les « affiche » ou réalisations produites pourront être modifiées, seront propres et lisibles. Les objectifs que visent l'utilisation de l'ordinateur comme affiche évolutive est de faciliter la créativité du groupe, permettre la communication entre les groupes et à l'intérieur du groupe. Pour que ce type d'activité soit efficace il faut constituer des petits groupes de quatre (4) apprenants maximum et chaque groupe est connecté à un réseau grâce à son ordinateur.

- **Outil de laboratoire dynamique** : L'introduction des ordinateurs au laboratoire simplifie la préparation et la surveillance des expériences, mais elle donne surtout la possibilité de présenter, simultanément le phénomène véritable et une reproduction graphique de celui-ci. C'est ce type innovant d'apprentissage, de l'abstrait par le concret, du symbolique par le réel, qui a été rendu possible grâce à ordinateur. Cet apprentissage se fait en présentant en temps réel, une représentation graphique de l'interaction des variables du phénomène à l'étude durant le déroulement du phénomène physique lui-même. C'est cette utilisation pédagogique de l'ordinateur qu'en leurs temps, certains auteurs ont désigné par la métaphore « Lunette cognitive ». C'est une acquisition de l'abstrait par le palpable. L'intention didactique à long terme étant de retourner cette séquence dans le but d'attribuer à ce graphique irréel les mêmes spécificités qu'un autre identique plus habituel et plus connu de l'étudiant que les phénomènes qu'il devra étudier. C'est uniquement au moment où le graphique aura un sens pour l'élève que celui-ci deviendra un véritable outil cognitif qui lui sera disponible pour appréhender de nouvelles connaissances. C'est la condition sine qua none pour que le laboratoire dynamique, en profitant du langage de codage graphique et mathématique, devienne une situation didactique privilégiée pour l'acquisition d'un véritable savoir-faire expérimental (Bénézra, Jean & Rothan, 1989).

- **Évaluateur instantané** : Dans le cadre d'une utilisation pédagogique de l'ordinateur en tant qu'évaluateur instantané, le but est d'établir un diagnostic relativement à des objectifs donnés. Il s'agit de corriger une production de l'apprenant de manière objective, fiable, automatique, immédiate et surtout sans intervention de l'enseignant (Bénézra, Jean & Rothan, 1989). Pour cette utilisation pédagogique, l'ordinateur corrige instantanément les exercices des apprenants, ce qui offre à l'enseignant la possibilité de poser un diagnostic afin de mettre en place une pédagogie différenciée, l'enseignant peut aussi être aidé dans la gestion de la classe par la décharge d'une partie des tâches de correction. Dans la situation ou plusieurs groupes d'apprenants travaillent en parallèle, chacun à son rythme, sur des sujets différents ou identiques, l'ordinateur utilisé de cette manière va assister l'enseignant dans les travaux de correction (Bénézra, Jean & Rothan, 1989).

- **Répétiteur « inlassable »** : Il est également envisageable, dans le but d'approfondir ou de réviser une notion repérée en classe, d'orienter des élèves vers un travail en libre-service ou pendant une durée déterminée à l'aide d'un ordinateur. Utiliser l'ordinateur comme répétiteur vise prioritairement à consolider une notion, grâce à un entraînement basé sur la répétition. Ces dernières peuvent être validées ou non, corrigées ou non, mais rarement expliquées dans une réelle perspective de remédiation. Le répétiteur peut également constituer un outil d'évaluation, grâce au traitement informatique, sous réserve que les réponses attendues ne soient pas ouvertes. Les compétences mises en œuvre sont plutôt d'ordre disciplinaire (Bénézra, Jean & Rothan, 1989). Il existe diverses applications conçues pour exercer l'apprenant : il y a les fiches d'exercices, les exercices répétés (drills) et les environnements. Les activités cognitives que les élèves mettent en œuvre dans cet usage sont souvent de reconnaître des items, de produire une réponse, de recevoir une rétroaction de l'ordinateur. En revanche, il convient déjà de signaler que cet usage de l'ordinateur ne peut se faire totalement hors du contrôle de l'enseignant dans la mesure où l'enseignant a le devoir de tester l'application avant son emploi par les élèves. En plus, au cours du travail, l'enseignant doit suivre d'assez près la progression des élèves.

- **Tuteur interactif** : Dans les années cinquante, sous l'impulsion de la cybernétique et de la psychologie comportementale, une technologie de l'enseignement s'est implantée, c'est-à-dire l'application des méthodes scientifiques et des connaissances sur les processus d'enseignement en vue d'atteindre des objectifs éducatifs précis et contrôlables. Si initialement, l'ordinateur ne servait qu'à être un support à l'enseignement programmé, il s'est transformé plus tard en une machine adaptative à l'élève grâce notamment aux techniques de l'intelligence artificielle. On pourra alors utiliser l'ordinateur comme tuteur non seulement interactif, mais aussi « intelligent ». La conception de telles applications est une tentative pour concevoir un programme tournant sur un ordinateur capable de se conduire d'une telle façon qu'elle serait jugée comme un « bon enseignement » si elle était faite par une personne humaine, c'est-à-dire concevoir un programme ayant la triple expertise : celle du domaine à enseigner, celle de l'enseignement et celle de l'analyse des compétences et connaissances correctes et erronées des élèves. Nous voyons donc que si l'on met à la disposition de l'enseignant un tel outil, il apportera à l'apprenant des connaissances nouvelles (ou assoira les anciennes). Dans un contexte de pédagogie differenciée, l'enseignant fait travailler sur l'ordinateur une partie de la classe sur une notion non encore aborbée. Il peut ainsi se consacrer aux differents groupes constitués : explorer, manipuler, comprendre par anticipation la nouvelle notion (non encore abordée en classe) pour le groupe qui a réussi la partie actuellement étudié ; réviser, revisiter et s'exercer sur la notion présentement étudiée pour le groupe en difficulté. Le travail en binôme, avec une machine par binôme, accompagné de consignes de procédures, permet une

meilleure appropriation des connaissances grâce au dialogue qui s'établit entre les élèves, nécessitant argumentation et explication.

Au terme de cette présentation, il est important de signaler que les six utilisations pédagogiques de l'ordinateur que nous venons de décrire ne sont pas les seules qui existent. Plusieurs autres typologies ont été faites par d'autres chercheurs. Il existe des classifications dites anciennes qui présentent les utilisations pédagogiques de l'ordinateur dans deux environnements : un environnement fermé dans lequel on trouve les enseignements assistés par ordinateur (EAO) et un environnement ouvert caractérisé par les apprentissages assistés par ordinateur (AAO), et des classifications plus ressentes. Dans ces dernières, les utilisations pédagogiques de l'ordinateur ont été faites selon cinq (5) situations d'apprentissage. Les auteurs de cette classification Bénézra (1989) ; Jean & Rothan (1989) envisagent de cette manière l'ensemble des possibilités qu'offrent les ordinateurs et la technologie en général. Cette classification repose sur les situations d'apprentissages suivantes :

1- la génération d'écrits et ses exploitations ;

2- les images mentales, l'articulation des connaissances individuelles et collectives ;

3- la résolution de problèmes, l'exécution de projets personnels ;

4- la consultation de ressources et sources de références ;

5- l'acquisition de connaissances spécifiques et la gestion de l'édification de ces connaissances.

Le but visé ici n'étant pas de présenter toutes les typologies, même pas d'en faire une étude exhaustive ni comparative, nous voulions autant que faire ce peu en présenter une, celle qui nous semble la plus complète, la plus facile à cerner et qui cadre le mieux avec notre contexte. Dans cette perspective, nous constatons que dans la première classification, le logiciel d'aide à l'élaboration d'algorithmes s'inscrit dans le cadre de l'utilisation de l'ordinateur comme encyclopédie active et dans la seconde, il s'agit de la résolution de problèmes. Lorsque l'on simule un algorithme avec AlgoBox, il ne s'agit pas de permettre à l'apprenant de voir s'exécuter de façon virtuelle un phénomène physique réel (comme voire en quelques minutes la croissance d'une plante). On serait dans une situation d'utilisation de l'ordinateur comme outils de laboratoire dynamique, mais AlgoBox permet d'exécuter un algorithme pour voir s'il résout bien le problème pour lequel il a été établi, s'il donne bien le résultat escompté. Il s'agit donc de simuler un algorithme qui lui-même n'est pas un phénomène physique mais un langage codifié (ensemble de symboles respectant une certaine norme) qui doit donc véhiculer un message. La simulation aura donc pour ambition de voir sur l'écran d'un ordinateur si le message contenu dans la codification que nous avons faite est celui souhaité donc donne le résultat attendu. Dans le cadre de notre recherche, nous souhaitons établir que la difficulté d'apprentissage en algorithmique réside dans la double activité cognitive que mène l'apprenant dans la résolution d'un problème en algorithmique. En prenant donc

en charge toute la partie liée au formalisme et vérifiant par la simulation si le problème est résolu, AlgoBox facilite l'apprentissage en ne confiant à l'apprenant que la seule activité de résolution du problème. Cependant, son interface pas très attrayant, l'absence de possibilité multimédia, notamment le son, représentent de sérieuses limites lorsqu'on se projette dans la perspective d'un apprentissage multimédia au sens de Mayer (2001).

La mobilisation à des fins d'enseignement, des technologies interactives de communication et de traitement automatique de l'information suggère des possibilités importantes d'adaptation à cette augmentation des besoins de formation. En effet, deux des caractéristiques des utilisations éducatives de l'ordinateur permettent de lever des contraintes fondamentales dans les processus d'enseignement : celle du lieu et celle du temps, qui condamnent les systèmes de formation classiques à fonctionner dans le cadre et les limites de systèmes scolaires.

## 2.1.4. Autres utilisations pédagogiques de l'ordinateur

L'apport décisif de l'ordinateur est de permettre de concevoir, d'organiser et de promouvoir à coté des enseignements traditionnels, des systèmes complémentaires ou alternatifs d'enseignement-apprentissage. Au nombre de ces systèmes alternatifs ou complémentaires, on citera entre autre l'enseignement à distance, l'enseignement assisté par ordinateur et l'accès à des banques de données. Tout ceci allant dans le sens d'une multiplication des sources et des supports d'informations pour représenter et simuler les savoirs... Les caractéristiques de ces environnements leur permettent de s'adapter à la variété des besoins de formation ; elles répondent ainsi à une demande sociale importante. Ce versant pragmatique ne doit pas se désintéresser du fait que c'est de beaucoup à cause de la recherche fondamentale que de tels environnements ont pu être développés. Ainsi, toutes les disciplines concernées par la représentation, la gestion et la transmission des connaissances comme l'informatique, l'intelligence artificielle, la psychologie cognitive, les sciences de l'éducation, la didactique, l'épistémologie concourent naturellement à la résolution des défis technologiques et conceptuels qu'elles n'auraient pas pu aborder seules. Ce nouveau champ de recherche tente de répondre aux questions qui se posent lorsque l'on cherche à appliquer les technologies multimédia (vidéo, télématique, informatique, intelligence artificielle...) à des tâches d'enseignement. Il recouvre de ce fait deux grands thèmes. Le premier est explicitement orienté vers la technologie et représente les efforts de mise en commun des différents savoir-faire de chaque discipline pour aboutir à l'essor des systèmes informatiques. Le second concerne les retombées de ces efforts dans le cadre de la recherche fondamentale dans chacune des disciplines concernées. Il peut exister effectivement un risque de poursuivre des objectifs qui peuvent paraître incompatibles. Cela pourrait réduire a priori les chances d'obtenir des résultats satisfaisants. Toutefois, une analyse plus précise révèle que cette ambiguïté correspond au paradigme fondamental de ce champ de recherche : « un logiciel d'enseignement est à la fois un outil concret d'apprentissage et la formalisation d'une théorie de l'apprentissage » (Bierman, 1987). Les travaux des chercheurs illustrent cette interaction dynamique où la recherche

fondamentale se confond à la recherche appliquée : l'implémentation et l'expérimentation d'un environnement d'apprentissage renvoient au concepteur des questions fondamentales de recherche, lesquelles génèrent de nouveaux systèmes. Wenger (1987) abonde dans ce sens lorsqu'il déclare : « le fait que la conception de tutoriels intelligents requiert une aussi profonde compréhension des processus en jeu pourrait signifier que nous avons trouvé une méthodologie au moyen de laquelle nous pouvons nous attaquer à des questions générales de manière systématique ».

Cependant, on peut se demander quelle est la bonne utilisation de l'ordinateur à l'école. Le bon usage est celui qui saura améliorer considérablement les pratiques pédagogiques. Identifier et utiliser cet apport dans le processus enseignement-apprentissage est la question centrale.

Dans les différentes utilisations pédagogiques de l'ordinateur, plusieurs outils sont mis à la disposition des acteurs de l'école, parmi ces outils, il y a les didacticiels. Dans la diversité des didacticiels, certains sont destinés à un usage très ponctuel alors que d'autres s'inscriront dans la durée. Notons que tout didacticiel présente au moins un intérêt, utilisons-le pour ce qu'il apporte, en prenant acte de ses limites, voire de ses défauts qui ne sont d'ailleurs pas toujours dissuasifs. Quand bien même un logiciel serait irréprochable, il ne fera jamais de miracle, car ce n'est pas le logiciel qui sera déterminant, mais l'usage qu'on en fait.

Les didacticiels ne sont pas les seuls outils informatiques d'apprentissage utilisés dans l'enseignement. L'école, dans sa quête d'intégration de nouveaux outils informatiques d'apprentissage utilise également:

- les plateformes d'apprentissage en ligne qui sont des sites internet (web) qui hébergent des contenus didactiques.

- Les espaces numériques de travail ou d'apprentissage (ENT) ou (ENA) qui sont des portails sécurisés sur internet qui donnent la possibilité aux élèves, parents, enseignants et personnels administratifs de partager des travaux collectifs ayant un lien avec l'activité d'éducation et d'accompagnement des élèves.

- Les tableaux blancs interactifs (TBI) qui sont des dispositifs associant les avantages d'un écran tactile et de la vidéoprojection.

- Les tablettes interactives représentent des ordinateurs mobiles, très légers et maniables disposant d'écran tactile.

- A l'instar de Scratch, il y a les logiciels de développement qui permettent de développer la créativité, la curiosité intellectuelle et le goût d'apprendre.

- Les réseaux sociaux qui servent à la diffusion de l'information tant chez les élèves que chez les parents. L'enseignant peut aussi y diffuser sa planification hebdomadaire.

- Les Glogsters sont des outils de travail intéressant sur le net, ils permettent tant aux enseignants qu'aux élèves de créer des affiches personnalisées: des Glogs, dans lesquelles il est possibles d'insérer du texte, des images, des photos, des vidéos, du son, des dessins, des pièces jointes, des effets spéciaux, etc., le Glogsters est un puissant outil multimédia qui permet en plus d'interagir avec son auditoire.

- Pearltrees quant à lui est un service web qui sert à choisir, organiser et partager des contenus numérique.

- Enfin, les outils de robotique pédagogique donnent la possibilité aux élèves de construire des machines (robots) et de les faire fonctionner en se servant d'une application développée pour cet usage. Les activités de robotique pédagogique visent la stimulation de la créativité et la réalisation, la réflexion dans le but de résoudre de manière créative et alternative des problèmes et l'apprentissage de la communication, du partage d'idées et du travail collectif.

On ne peut décemment se prononcer contre les utilisations pédagogiques de l'ordinateur, dès lors que l'on admet qu'ils ne constituent qu'un des outils au service de la pédagogie. Bien au-delà de l'outil proprement dit, c'est l'activité que l'outil engendre qui doit avoir un sens et apporte une plus-value dans la pratique pédagogique, grâce à des objectifs et des stratégies formulés de manière univoque. Il appartient donc aux enseignants de prendre conscience, à travers leur pratique professionnelle, des véritables enjeux pédagogiques liés à l'utilisation de l'outil informatique. Ces enjeux peuvent être concourants ou aux antipodes des discours institutionnels car tout discours, aussi officiel soit-il, ne saura remplacer le temps, la volonté d'expérimenter, la formation continue et l'aide sur le terrain, autant de conditions indispensables à l'intégration efficiente de l'ordinateur dans le processus enseignement-apprentissage.

La simulation est au centre de notre utilisation de l’ordinateur dans l’apprentissage de l’algorithmique, non pas au sens de l’observation virtuelle d’un phénomène physique réel, mais dans matérialisation d’une idée et la mise en œuvre du processus de réalisation de cette idée pour voir si elle donne le résultat escompté.

Que pouvons-nous alors dire de certains outils de simulation d’algorithme actuellement utilisés à l’école, ainsi que sur celui que nous avons utilisé pendant notre expérimentation ?

# 2.2. Outils de simulation d’algorithmes

## 2.2.1. Cadre général des outils de simulation

Il est courant de nos jours d'observer que les logiciels et plus généralement les artefacts numériques ont une incidence sur l'enseignement. L'incidence de la technologique et de l'informatique sur l'enseignement s'observe notamment par la simulation, une simulation qui offre la possibilité d'étudier et de se familiariser avec des

situations qui ne sont pas directement accessibles à une résolution exacte ou approchée, tout au moins au niveau d'enseignement considéré (Wilensky, 2003).

En France, c'est à travers les statistiques que la simulation a réellement fait son entrée dans l'enseignement secondaire, avec les programmes du lycée en seconde. Les calculatrices et les tableurs ont été les deux technologies utilisées principalement à cet effet et les documents qui accompagnent les programmes récents tentent d'outiller les enseignants face à cette importante révolution dans l'enseignement des statistiques.

Une simulation est une image ou un modèle d'un événement, d'un mécanisme ou phénomène imaginé pour présenter et faire comprendre le fonctionnement d'un système (Parrochia, 2000).

Le concept de simulation est initialement apparu dans le domaine de la recherche comme une technique qui permet d'étudier les résultats d'une action sur un phénomène sans devoir intervenir sur le phénomène réel. On se sert des simulations pour comprendre les principes de fonctionnement d'un certain type de processus physiques, biologiques et sociaux.

Bien que la simulation ait existé avant l'apparition de l'ordinateur, les simulations informatiques, basées sur des algorithmes souvent très complexes, sont très largement les plus diffusées. Elles constituent des environnements dans lesquels les apprenants manipulent des composants d'un système de manière hautement interactive. Elles permettent d'étudier le fonctionnement et les propriétés d'un système modélisé et de prédire son évolution. Les interfaces graphiques des ordinateurs permettent de construire des simulations très réalistes sur la base d'images de synthèse qui reflètent d'une manière souvent très fidèle la réalité (Parrochia, 2000).

Sorte d'expérience, outil intellectuel ou théorique d'analyse, ou bien située entre théorie et expérience et source d'information sur la nature des choses, la disposition épistémologique des simulations est un sujet de discussion parmi les scientifiques et les épistémologues (Parrochia, 2000). Bien que son statut épistémologique fasse débat, la simulation peut, dans l'enseignement scientifique, fonctionner comme une passerelle entre théorie scientifique et monde réel et être assujetti à l'expérimentation. Dans une perspective plus constructive, il serait plus intéressant de débattre non pas sur le statut mais sur la fonction de la simulation. Les missions de la simulation peuvent changer selon le domaine concerné et dans chaque cas. On peut ainsi :

- simuler pour comprendre ;
- simuler pour concevoir ;
- simuler pour agir.

L'approche sociologique, quant à elle met en exergue l'élaboration sociale de la preuve. L'épistémologie de Cartwright (1983; 1999) tient compte d'une manière

originale, l'activité scientifique, elle inclut l'aspect social de la simulation et elle hisse l'importance de l'expérience, de la situation et des instruments.

Il pourrait être question de mieux connaître les utilisations scolaires de l'enseignement de la simulation, et son impact sur les apprentissages, au-delà des finalités qui peuvent se résumer en : apprendre à expérimenter et à observer, à établir des cohérences d'actions ou d'actes mentaux exercés sur des données concrètes ou sur des codes. Ainsi, depuis plusieurs années, de nombreuses études avaient comparé les effets sur l'apprentissage d'un enseignement s'appuyant sur une simulation, par rapport à un enseignement transmissif à l'aide d'exposé. Ces études ont été réalisées dans des domaines comme la biologie, la mécanique, ou bien encore l'électricité.

Certains travaux, à l'instar de ceux effectués par (de Jong et al., 1999), montraient que l'apprentissage à l'aide de la simulation était plus agissant dans un environnement qui donnait des éventualités de recherche et de découverte. Le cadre informatique spécial servant à la formation en mécanique, contient l'enregistrement automatique des itinéraires des étudiants. Ces données, corrélées à l'analyse des commentaires de ces derniers peuvent concourir à comprendre leurs raisonnements lors des différents épisodes.

Les recherches françaises ont tendance à s'intéresser un peu plus aux procédures d'enseignement ou de médiation ayant recours à de la simulation. En effet, ces recherches montrent ainsi l'importance du contexte d'enseignement ou d'apprentissage et la nécessité d'une médiation adaptée, en particulier pour les plus jeunes élèves.

De nos jours, le recours à des outils de simulation en situation d'apprentissage est très habituel, tant dans le cadre de l'enseignement général, technologique ou professionnel, au secondaire comme au supérieur. La simulation offre à l'apprenant un lieu d'investigation, elle s'appuie sur des environnements interactifs dans lesquels le plaisir est associé à la découverte scientifique et technologique et ceci de manière active. L'apprenant, en interaction avec l'environnement matériel et humain, développe la curiosité, le questionnement, l'observation, le tâtonnement expérimental. Toutes ces démarches étant prescrites par les programmes scolaires pour les sciences et les techniques.

La simulation est indispensable dans les expérimentations pour concrétiser des phénomènes naturels ou de systèmes techniques, impossibles à présenter sous forme réelle. L'impossibilité d'observer ces phénomènes provient de plusieurs paramètres, il peut s'agir d'une échelle de temps, qui est extrêmement rapide ou lente ou ne concorde pas avec la durée de l'apprentissage. Il peut aussi être question des dimensions qui rendent les phénomènes impossibles à reconstituer dans l'espace scolaire. Il peut enfin s'agir de la nature qui n'est tout simplement pas compatible avec les exigences inhérentes à de tels environnements. En sciences de la vie et de la terre, par exemple, peu de réactions biologiques sont observables l'espace d'une séquence pédagogique

parce qu'elles sont immédiates ou interminables pour qu'elles soient regardées en temps réel.

La mise en œuvre de démarches de développement de la simulation en classe suppose de mieux connaître les usages qu'en font les enseignants et les exigences ou contraintes d'utilisation pour des apprentissages.

## 2.2.2. Cadre méthodologique des outils de simulation

Les outils de simulation informatiques peuvent être utilisés en adéquation avec les idées de l'apprentissage inspirées du constructivisme et plus particulièrement avec le modèle du compagnonnage cognitif qui prône l'apprentissage à travers la confrontation avec des situations réaliste. Ces outils sont des systèmes permettant à l'apprenant d'utiliser un modèle pour comprendre le fonctionnement d'un dispositif technique ou d'un phénomène naturel et, s'appuyant sur les observations faites, de se construire une représentation du dispositif ou la notion étudiée.

Dans l'enseignement par la simulation, l'objectif pédagogique principal est que les élèves construisent des savoirs en simulant des phénomènes scientifiques. Les cadres théoriques et méthodologiques requis peuvent être multiples. Dans le model-based learning de la littérature anglophone, il est question de faire progresser les modèles mentaux des apprenants vers des modèles plus scientifiques. Dans ce cadre théorique, un aspect essentiel de l'apprentissage des sciences a pour but de faire construire par les apprenants de modèles mentaux qui ressemblent le plus possible aux modèles scientifiques et historiques conventionnels (Hodson, 1992). Ces différents modèles, aussi bien mentaux que scientifiques, ne sont certainement pas de la même espèce, registre psychologique pour les uns, épistémologique pour les autres. On se demande alors comment peut s'envisager le passage de l'un vers l'autre.

Les versions réduites des modèles scientifiques, dérivées de ces études, peuvent être utilisées pour les programmes scolaires et être appelées « modèles enseignés » (Gilbert, Boulter & Elmer, 2000). Pour Clément (2000), cette théorie de l'apprentissage s'inscrit dans le développement d'une théorie du changement conceptuel. L'emploi de la simulation en classe peut également faire revisiter la perception que l'on a sur la notion d'étayage ou échafaudage (scaffolding). Rappelons que l'étayage, provient du modèle socioconstructiviste vygotskien de l'apprentissage, il doit permettre à l'apprenant de pratiquer une activité cognitive au-delà des capacités acquises. Il intervient dans une situation d'interactions sociales (Vygotsky, 1978). Vergnaud (2000) explique que les processus de médiation dans la théorie de Vygotsky sont bidimensionnels mais non indépendants : la médiation par l'adulte et la médiation par les signes.

Dans le contexte scolaire, le rôle d'accompagnateur revient souvent à l'enseignant. Avec les utilisations scolaires des technologies informatiques, une nouvelle question est apparue : savoir si la simulation peut contribuer à cet étayage.

Cependant, il est essentiel de déclarer que les systèmes de simulation sont immensément utilisés en éducation et en formation à cause de leur capacité à présenter aux apprenants des informations hautement réalistes. En particulier, le recours à la simulation en éducation est inévitable lorsqu'il n'est pas possible de réaliser l'expérience réelle au regard de son caractère préjudiciable à l'éthique, trop couteuse ou trop longue.

Une simulation éducative est basée sur la modélisation d'un phénomène, d'un dispositif que l'élève apprend à maitriser en interagissant avec la simulation (Alessi et Trollip, 2001).

Il est à noter qu'une simulation n'est pas une réplique exacte d'un phénomène : elle est le plus souvent une simplification de celui-ci en omettant, changeant ou augmentant des particularités ou des caractéristiques. En se servant des simulations qui s'appuient sur des modèles simplifiés ou qui proposent des simulations dont le modèle se complexifie au fur et à mesure que l'élève avance dans son apprentissage. Les simulations pédagogiques permettent de confronter les apprenants à des phénomènes qu'ils apprendront à maitriser progressivement. Parfois, les simulations éducatives ajoutent au modèle des éléments qui n'existent pas dans la réalité, dans ce contexte on parle de « réalité augmentée ». Il s'agit par exemple, de forme de tutorat qui procurent à l'apprenant des rétroactions ou qui fournissent des suggestions pour l'aider à comprendre des phénomènes ou des mécanismes complexes observés à partir d'un logiciel de simulation.

Alessi et Trollip (2001) proposent une catégorisation des systèmes informatiques de simulation :

- les simulations conceptuelles, elles reposent sur le principe d'un système qui peut être soit naturel ou artificiel, c'est-à-dire les simulations « à propos de quelque chose ».

- les simulations opérationnelles, ce sont des simulations qui accomplissent un modèle de processus, c'est-à-dire qui traitent de « comment faire quelque chose ».

Les simulations conceptuelles comportent des principes, des concepts et des faits relatifs à un système qui doit être simulé. Cette catégorie de simulation peut elle-même être classée en deux catégories :

  - les simulations physiques, pour lesquelles un objet est représenté à l'écran et offre à l'apprenant la possibilité d'apprendre sur cet objet ;

  - les simulations itératives. Dans ce type de simulation, le temps ne constitue pas en général une variable du système simulé. L'apprenant ne manipule donc pas le temps réel pour observer un phénomène, il exécute la simulation à plusieurs reprises en modifiant chaque fois certains paramètres.

Les simulations opérationnelles incluent des séquences d'opérations cognitives ou non cognitives (des procédures) qui peuvent être appliquées au système simulé. On peut les classer en deux catégories : les simulations procédurales et les simulations situationnelles.

Une simulation procédurale permet de monter une séquence d'actions afin d'atteindre un but ou pour effectuer une tâche (simulateur de vol). En revanche, une simulation situationnelle représente des comportements et des attitudes d'une population ou d'une organisation dans différentes situations. La plupart des simulations dites situationnelles incorporent des jeux de rôles (Alessi et Trollip, 2001).

Une forme particulière de simulation connait aujourd'hui un développement considérable : il s'agit des simulations ludoéducatives désignées par exemple par l'expression « jeux vidéo ». Les jeux vidéo se transforment en systèmes informatiques très complexes d'un point de vue graphique, d'interactivité et de narration. La variété des jeux existants est très étendue, tant sur le plan du matériel (consoles spéciales, carte graphique, périphériques spécialisés) que des logiciels et des thèmes traités. Selon Malone (1981), les jeux vidéo disposent d'une grande capacité pour motiver leurs utilisateurs. Cette motivation, très importante en situation d'apprentissage, est basée sur trois aspects : l'imagination, la compétition et la curiosité. Dans un contexte éducatif, les jeux vidéo ont la possibilité de stimuler l'intérêt des élèves pour l'apprentissage et accroître leur capacité de rétention (Prensky, 2001). La plupart des jeux vidéo sont des applications excessivement interactives qui requièrent de la part de ceux qui s'en servent un certain type de fonctionnement, la mémorisation, l'anticipation ainsi que l'élaboration et de rapports spéciaux et la mise en place de stratégies.

Les jeux vidéo exigent des potentialités cognitives réelles et certains d'entre eux peuvent même être utilisés comme des outils à fort potentiel cognitif. Une question récurrente se pose cependant sur la manière dont les élèves s'en servent. Construisent-ils de nouvelles connaissances en analysant des données, en testant des idées et en vérifiant des hypothèses ou manipulent-ils simplement le jeu à un niveau superficiel en cliquant sur des boutons et en sélectionnant des options pour voir ce qui se passe ? De plus, dans le contexte de notre recherche, où nous avions fait le constat d'une très forte démobilisation et un fort désintérêt face au caractère théorique du cours d'algorithmique, ne pourrait-on pas envisager l'apport d'une simulation ludoéducative ? L'apprentissage par l'action et par la découverte, et l'expérimentation active que les jeux vidéo (principalement les jeux de simulation) conduisent à mettre en place, constitue un aspect très important qui valorise l'usage des jeux à des fins éducatives dans une approche constructiviste.

En règle générale, les jeux mobilisent et développent des compétences qui relèvent de la logique, de la mémoire et de la résolution de problèmes, toutes compétences visées aussi par l'enseignement de l'algorithmique, mais aussi, pour certains jeux, de la pensée critique et de la recherche. Leur utilisation requiert des joueurs qu'ils manipulent des

objets à l'aide d'outils technologiques et développent ainsi une compréhension du jeu en tant que système complexe. Les jeux qui se jouent en collaboration ou en coopération favorisent le développement d'aptitudes sociales telles que décider en commun, définir et se mettre d'accord sur des buts. Pour pratiquer certains jeux de manière efficiente, il faut être en mesure d'effectuer des traitements parallèles de données, c'est-à-dire de tenir compte au même moment de plusieurs sources d'informations. Dans ce contexte, les jeux vidéo peuvent soutenir le développement d'une série de comportements, comme l'élaboration de stratégies, la planification, la communication, la manipulation des chiffres, les aptitudes de négociation, la prise de décision collective et la manipulation de données.

## 2.2.3. Quelques outils de simulation

Sans avoir la prétention d'être exhaustif, nous voulons dans ce point présenter quelques applications utilisées en milieu scolaire pour enseigner l'algorithmique. Ces applications sont constituées de logiciels dédiés, de logiciels de calculs formels et enfin de logiciels de programmation.

Les logiciels de calculs formels sont des applications qui facilitent le calcul symbolique. Ces applications ont pour principal ambition la manipulation des expressions mathématiques sous leur forme symbolique. Le calcul formel, ou parfois appelé calcul symbolique, est le champ des mathématiques et de l'informatique dont l'objet d'études se trouve être les algorithmes qui s'appliquent sur des objets de nature mathématique par l'intermédiaire de représentations finies et exactes. Ainsi, un nombre entier sera représenté de manière finie et exacte par la suite des chiffres de son écriture binaire (base 2). Si nous considérons les représentations de deux nombres entiers, le calcul formel va donc s'intéresser par exemple à la question du calculer de leur produit. Il existe plusieurs logiciels de calculs formels donc on peut citer : MATHLAB, XCAS, MAXIMA, WIRIS...

Comme leurs noms l'indiquent, les logiciels de programmation sont des logiciels qui permettent à leur utilisateur de faire de la programmation (de programmer). La programmation est une activité qui consiste à écrire des lignes de code (commands en anglais) qui seront interprétées par l'ordinateur. L'objectif étant de développer une application utilisable et visible sur l'écran d'un ordinateur, recevoir, traiter et envoyer des informations par son biais. Afin d'aboutir à cette production, le programmeur se sert d'un langage de programmation. Le langage de programmation est une transcription conformiste dont l'objet est d'établir des algorithmes et parvenir à des programmes informatiques qui les mettent en œuvre. Par analogie avec les langues que nous utilisons au quotidien, un langage de programmation dispose comme ces dernières d'un vocabulaire et de règles syntaxiques (grammaire). Les langages de programmation permettent de décrire les structures des informations à traiter et de montrer comment les traitements se feront, selon quels algorithmes. Ces langages servent à mettre en rapport le programmeur avec l'ordinateur, ainsi que les programmeurs entre eux. Les

programmes sont d'ordinaire rédigés, consultés, compris et corrigés par plusieurs programmeurs travaillant en équipe.

Quant aux logiciels dédiés, ce sont des logiciels qui ont été développés pour un usage spécifique dans un domaine donné. Ils sont dévoués par exemple à un corps de métiers à une discipline scolaire. On parle de logiciel dédié à la comptabilité, logiciel dédié à la librairie, logiciel dédié à l'étalonnage, etc., dans notre cas, on parlera de logiciel dédié à l'algorithmique. Ces quelques exemples de ce dernier type de logiciels que nous allons présenter.

Il est à signaler que notre travail ne consiste pas à faire une étude des différents logiciels de simulation d'algorithmes, ni de faire une étude comparative ou critique de ces derniers. Nous voulons simplement dans les lignes qui suivent présenter quelques applications hormis celle qui a été utilisée pour notre expérimentation.

### 2.2.3.1. Execalgo[(1)]

Execalgo est un logiciel conçu par un groupe d'experts qui a pour objectif de faciliter la compréhension de la notion d'algorithme (fonctionne sous Windows).

C'est un logiciel gratuit orienté vers l'apprentissage de l'algorithmique puisqu'il a été développé pour les élèves de première L et Terminale L qui suivent l'option mathématiques. C'est une application pédagogique, les instructions sont écrites en langage naturel, elles sont pré-écrites sur la fenêtre du programme, on se contente de compléter à l'aide d'instructions chaque partie de l'ossature du programme.

Cependant, ce logiciel a un certain nombre de défauts qui paraissent rédhibitoires : il ne possède pas d'instruction « input », il gère mal les instructions conditionnelles (on doit pour cela se servir des points de branchements), et il ne dispose pas non plus de fenêtre graphique. En France où il a été développé, il semble abandonné, en général, au profit d'AlgoBox.

Execalgo sert à faciliter l'enseignement de la notion d'algorithme. Un algorithme mis en œuvre avec cette application apparaît sous la forme d'une succession d'instructions dont les mots clés sont en français, formant un texte lisible, facilitant ainsi son analyse. L'issue de l'exécution se présente aussi sous forme de texte. Ces textes ont la possibilité d'être copiés, collés, enregistrés, imprimés, transmis par réseau ou Internet comme n'importe quel autre texte. Le texte du programme et le texte résultant de son exécution sont présentés côte à côte par le logiciel. L'exécution de l'algorithme peut se faire en mode pas à pas, ce qui donnera dans le texte résultat l'effet de chaque instruction.

Dans le cadre scolaire, un enseignant peut décider de se servir de ce logiciel en groupe ou individuellement, en s'appuyant sur la complexité des algorithmes qu'il va

---

**(1)** Le **logiciel Execalgo** a pour objectif de faciliter la mise en application sur ordinateur de l'algorithmique telle qu'elle est attendue notamment dans le nouveau programme de seconde**. (Téléchargeable à partir de l'URL suivant : https://www.ac-paris.fr/portail/jcms/p1_231670/execalgo-logiciel-pour-l-algorithmique).**

suggérer aux apprenants ainsi que des capacités mobilisables par ces derniers. L'enseignant peut aussi s'appuyer sur la disponibilité des équipements de l'établissement. Les compétences recherchées sont :

- Comprendre ce qu'est une variable et le type d'une variable.
- Comprendre comment les affectations ou demandes permettent d'agir sur les variables.
- Comprendre ce que sont les affichages, ou sorties.
- Comprendre la notion de déroulement séquentiel d'un programme.
- Comprendre les exécutions conditionnelles et les boucles.

Lorsque nous voulons, à l'aide de cette application écrire par exemple l'algorithme d'Euclide avec division et reste, il faut dans un premier temps décrire l'algorithme par un schéma ou un texte en français. L'algorithme d'Euclide peut être décrit par l'organigramme de la figure 2 ci-dessous, pour lequel les codages standards ont été utilisés : des rectangles pour les actions et un losange pour le test.

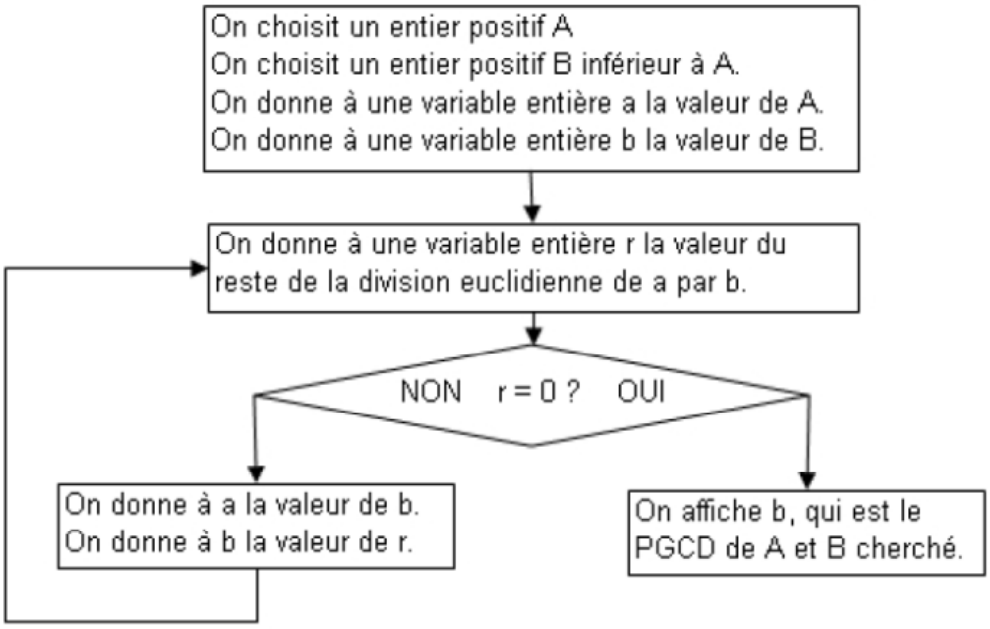



*Figure 2 : organigramme de l'algorithme d'Euclide sur Execalgo*

Ce schéma est déjà un premier pas vers la mise en application de l'algorithme : des variables sont nommées, les actions à effectuer, avant le test et selon son résultat, sont décrites avec précision. Puis, le schéma est traduit par une suite d'instructions. Cette étape oblige à se passer des avantages du schéma contenant les flèches susceptibles de partir dans toutes les directions pour indiquer l'instruction à exécuter après une instruction précise pendant le déroulement de l'algorithme. En effet, implicitement, les instructions sont exécutées selon l'ordre de saisie. Lorsque l'ordre d'écriture ne doit pas

être respecté, ces flèches sont remplacées par des instructions appelées ici « branchement conditionnel » lorsqu'elles dépendent du résultat d'un test et « branchement simple » dans le cas contraire.

Il faut d'autre part définir la partie du programme vers laquelle le branchement doit s'effectuer, c'est le rôle d'un « point de branchement ».

Notre exemple énoncé plus haut nous donne l'implémentation suivante:

| **Lignes de programme** | **Commentaires** |
|---|---|
| # Déclaration des variables | Une ligne commençant par # est une ligne de commentaire non analysée, n'entrant pas dans le décompte des instructions exécutées. |
| Variable(entier ; A ; B ; a ; b ; r ) | Types de variables possibles :<br>entier, réel, complexe, booléen, texte<br>On peut déclarer une liste, par exemple L(1..1000)<br>On peut déclarer un tableau : T(1..10,1..100) |
| # Entrées | Commentaire |
| Demander( A ) | Affectation choisie par l'utilisateur.<br>Syntaxe : Demander ( variable ) |
| Demander (B ) | Affectation choisie par l'utilisateur |
| # Initialisations | Commentaire |
| Affecter(a ; A) | Affectation définie par le programmeur<br>Celle-ci a pour objectif de conserver la valeur de A pour l'affichage final.<br>Syntaxe : Affecter ( variable ; valeur )<br>Voir la syntaxe plus détaillée en fin de document. |
| Affecter( b ; B) | Affectation définie par le programmeur |
| FaireTantQue ( r>0 ) | Boucle du type « Tant que »<br>Syntaxe : FaireTantQue(condition)<br>Instructions<br>FinFaire |
| Affecter ( r ; reste(a,b) ) | La fonction reste est prédéfinie, on peut ne pas l'utiliser et détailler le calcul en passant par le quotient (voir la syntaxe en fin de document). |
| Afficher (" a =" ; a; " b =" ; b ;" r = " ; r ) | Instruction d'affichage.<br>Voir la syntaxe en fin de document. |
| Affecter( a ; b) | Affectation |
| Affecter ( b ; r) | Affectation |
| FinFaire | Fait revenir à l'instruction FaireTantQue associée |
| Afficher ("Le PGCD de" ; A ; " et" ; B ; " est " ; a) | Affichage de sortie |

*Figure 3 : Exemple d'implémentation sur Execalgo*

Signalons que aussi bien le programme que son résultat sont présentés sous forme de texte simple comme dans la figure 4 ci-dessous et peuvent être enregistrés ou copiés et collés dans tout le document.

| Programme | Résultat de l'exécution |
|---|---|
| #Déclaration des variables<br>Variable ( entier ; A ; B ; a ; b ; r )<br>#Entrées<br>Affecter ( A ; 35786*7894)<br>Affecter ( B ; 4563*7894)<br><br># Initialisations<br><br>Affecter ( a ; max(A,B))<br>Affecter ( b ; min(A,B))<br>Affecter ( r ; reste(a,b))<br><br># Déroulement de l'algorithme<br>FaireTantQue(r<>0)<br>Afficher ( " a =" ; a; " b =" ; b ;" r = " ; r )<br>Affecter ( a ; b)<br>Affecter ( b ; r)<br>Affecter ( r ; reste(a,b))<br>FinFaire<br><br># Sorties<br>Afficher ( " " )<br>Afficher ( "Le PGCD de " ; A ; " et " ; B ; " est " ; b ) | a =282494684 b =36020322 r = 30352430<br>a =36020322 b =30352430 r = 5667892<br>a =30352430 b =5667892 r = 2012970<br>a =5667892 b =2012970 r = 1641952<br>a =2012970 b =1641952 r = 371018<br>a =1641952 b =371018 r = 157880<br>a =371018 b =157880 r = 55258<br>a =157880 b =55258 r = 47364<br>a =55258 b =47364 r = 7894<br><br>Le PGCD de 282494684 et 36020322 est 7894<br>Exécution terminée.<br>Instructions exécutées : 62 |

*Figure 4 : visualisation du programme et des résultats sur Execalgo*

Il faut aussi signaler que cette application possède un mode d'exécution dit « pas à pas », ce mode d'exécution permet de suivre le déroulement du programme établissant une correspondance entre la ligne de programme présentement exécuté et le résultat de cette exécution comme le montre la figure 5.

Exemple :

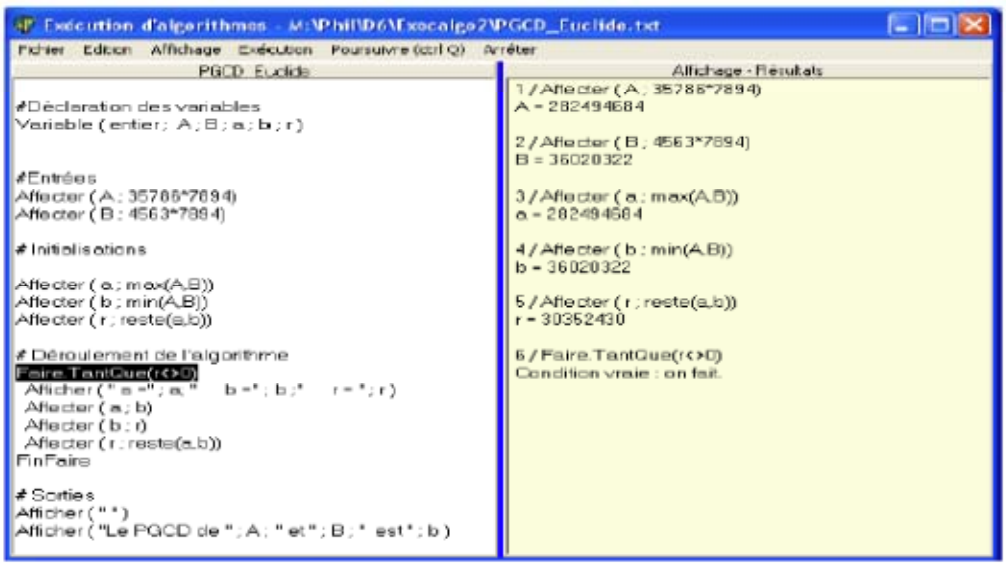



*Figure 5: visualisation en mode pas à pas sur Execalgo*

Comme toute application, Execalgo dispose d'un ensemble de menus qui contiennent des articles de menus (items) chacun remplissant une fonction spécifique de l'application. Execalgo dispose par exemple des menus suivants:

- Fichier ;
- Edition ;
- Instructions ;
- Fonctions ;
- Affichage ;
- Exécution.

Pour chacun de ces articles de menus, concernant les plus connus qui sont couramment rencontrés dans d'autres logiciels, les items y jouent le même rôle (copier, coller…). Par contre des menus comme « exécution » sont propre à ce type d'applications, ils ont des items comme « normal » ou « pas à pas » qui permettent l'exécution l'algorithme sous ces deux modes.

Execalgo utilise une syntaxe simplifiée qui permet comme dans beaucoup de logiciels dédiés de s'appuyer sur des règles très strictes. Ces algorithmes permettent l'usage de commentaires, de variables (réelles ou entières, des listes ou des tableaux), les affectations, les affichages, les exécutions conditionnelles, les modules et fonctions (sous-programmes), l'instruction « pause » des opérateurs arithmétiques, logiques et conditionnels.

### 2.2.3.2. Scratch[(2)]

Mis au point par le groupe de recherche Lifelong Kindergarten du laboratoire Média du Massachusetts Institute of Technology (MIT), Scratch est un langage de programmation qui simplifie l'imagination d'histoires interactives, de formes et objets animés, de jeux, de compositions de musique, de simulations par ordinateur et leurs partage sur le Web.

Scratch est une application en ligne (ou Online) conçue pour initier les élèves dès l'âge de huit (8) ans à des concepts fondamentaux en mathématiques et en informatique. Il s'appuie sur une approche ludique de l'algorithmique afin d'aider les apprenants à imaginer, à raisonner et à collaborer. Il n'est pas figé donne donc la possibilité de modifier le code du programme pendant l'exécution. Cette application multimédia dédiée à l'univers informatique des enfants traite facilement les concepts de base de la programmation comme les boucles, les tests, les assignations de variables, et surtout de

---

**(2) Scratch est un nouveau langage de programmation** qui facilite la création d'histoires interactives, de dessins animés, de jeux, de compositions musicales, de simulations numériques et leurs partage sur le Web**. (Téléchargeable à partir de l'URL suivant :** https://scratch.fr.uptodown.com/windows**).**

la manipulation des objets, tout comme les sons et les vidéos. Il favorise également leur partage sur le Web.

Scratch est visuel, cela signifie que tout le code est rédigé sans entremise dans la langue maternelle de l'enfant (une vingtaine de langues européennes sont disponibles). Le code de Scratch se compose de briques en couleurs qu'il faut agencer (par exemple les contrôles en jaune, les variables en orange, les mouvements en bleu). Scratch est libre permettant ainsi à l'enseignant de déployer sa pédagogie par le biais d'une interactivité quasi-ludique grâce à l'agencement de ces briques logicielles. Comme pour le mixage des sons, Scratch offre diverses possibilités comme celle de mélanger et de réutiliser des objets. Plusieurs offres interactives sont disponibles grâce à la Scratchboard qui dispose de capteurs de lumière, de son, de contacts, etc.

L'environnement Scratch se distingue des autres environnements de programmation par sa capacité à gérer la programmation événementielle voire parallèle : un projet Scratch ne se limite nullement à un unique algorithme, il inclut généralement des éléments multimédias (sons, images animées) ainsi qu'une multiplicité d'algorithmes s'exécutant tour à tour.

Ce logiciel offre plusieurs avantages:

- C'est un logiciel libre, gratuit, et multiplateforme.
- Il a été développé spécialement pour enseigner l'algorithmique et la programmation à des enfants.
- Il est très ludique.
- Les algorithmes sont exportables sur un site mutualisé.
- Il créé des animations, de petites vidéos et permet aussi d'y inclure des sons.
- La programmation consiste à agencer des petites briques sur lesquelles les instructions sont pré-écrites (pas de syntaxe à apprendre).

A contrario, il semble moins intéressant pour:

- Les couleurs très vives de l'écriture du code qui peuvent en rejeter d'autres.
- Il faut scinder toutes les opérations car il ne connait pas les règles de priorité.

Les projets Scratch sont faits d'objets**,** désignés aussi par le terme **«** lutins ». Ces derniers vont évoluer sur une scène. L'application permet de changer l'allure du lutin et aussi l'arrière-plan de la scène**.** Avec Scratch, il est possible de dessiner grâce à un éditeur de dessin, on peut aussi ouvrir et utiliser une image d'un disque dur, ou importer une image d'un site Web.

Il est aussi possible de donner des instructions à un lutin. Pour ce faire, on assemble des blocs de programmation afin de construire des piles. Ces piles seront appelées script lorsqu'elles seront surmontées d'un bloc de contrôle de type chapeau. Un clic sur un script déclenche l'exécution des blocs de programmation en partant du haut jusqu'en bas.

Comme nous venons de le voir, le principe de programmation de Scratch semble être plus proche de la programmation orienté objet en ce sens qu'il n'est pas nécessaire de rédiger une suite de codes complexes et en maitriser la syntaxe. Pour programmer avec Scratch, l'apprenant doit simplement déplacer des blocs de programmation, puis lancer leur exécution. Scratch dispose de trois (3) types de blocs :

- Les blocs de contrôle de type chapeau (figure 6): Ce sont les blocs de tête de pile, ils ont le dessus arrondi. Ils attendent un événement pour pouvoir exécuter les instructions correspondant aux blocs situés en dessous.


*Figure 6 : Exemple de bloc de contrôle de type chapeau*

- Les blocs de commande : Ce sont les blocs qui possèdent des bosses sur le fond et/ou des encoches sur le dessus comme illustré à la figure 7. Cette morphologie leur offre la possibilité d'être assemblé afin de constituer des piles. Certains blocs présentent une zone de saisie. Cette zone permet de taper un nombre ou choisir un article de menu. D'autres blocs, comme le bloc « Si.... Alors » ont un embranchement en forme de « C » offrant la possibilité d'insérer d'autres blocs dans la pile.


*Figure 7 : Exemple de blocs de commande*

- Les blocs à valeurs ou « Reporters » (figure 8): Ces blocs doivent être introduits à l'intérieur de la zone de saisie d'autres blocs. Ceux qui disposent d'extrémités arrondis renvoient des nombres ou des chaînes de caractères. On peut les introduire dans des blocs possédant des trous ronds ou rectangulaires. Les blocs à valeurs dont les bouts sont pointus renvoient des valeurs booléennes (vrai ou faux). On peut les insérer dans des blocs qui ont des trous avec des bords ronds ou pointus. Certains blocs à valeurs disposent d'une case à cocher à côté d'eux.


*Figure 8 : Exemple de blocs à valeurs*

Il est également important de signaler que Scratch dispose d'un éditeur de dessin pour dessiner de nouveaux lutins, de nouveaux costumes pour un lutin existant mais également d'autres fonds de scène (arrière-plans). Il a aussi une scène qui représente l'endroit où des histoires, des jeux et des animations prennent vie, les objets se déplacent interagissent les uns avec les autres. Deux modes de fonctionnement sont actifs sur Scratch : le mode présentation et le mode affichage.

### 2.2.3.3. Larp[3]

Développé initialement pour enseigner les concepts de la programmation structurée, LARP est acronyme qui vient de la compression de la phrase «Logiciel d'Algorithmes et de Résolution de Problèmes». C'est un langage de programmation qui offre la possibilité d'élaborer rapidement des algorithmes. Les programmes de LARP ont l'avantage d'utiliser des pseudo-codes à syntaxe flexible non compilable contrairement au code source. On formule ainsi des algorithmes en utilisant un langage semi-naturel plutôt que de devoir adhérer à une syntaxe rigide et cryptique telle que celle des langages de programmation traditionnels.

A coté de ce langage pseudo-code à syntaxe flexible, LARP offre aussi la possibilité de représenter les lignes de programme donc notre algorithme sous la forme d'un organigramme comme le montre la figure 9 ci-dessous.

```
\\ Pseudo-code simple
DÉBUT
  ÉCRIRE "Entrez un nombre"
  LIRE N

  SI N < 0 ALORS
    ÉCRIRE "Nombre négatif"
  SINON
    ÉCRIRE "Nombre positif"
  FINSI
FIN
```

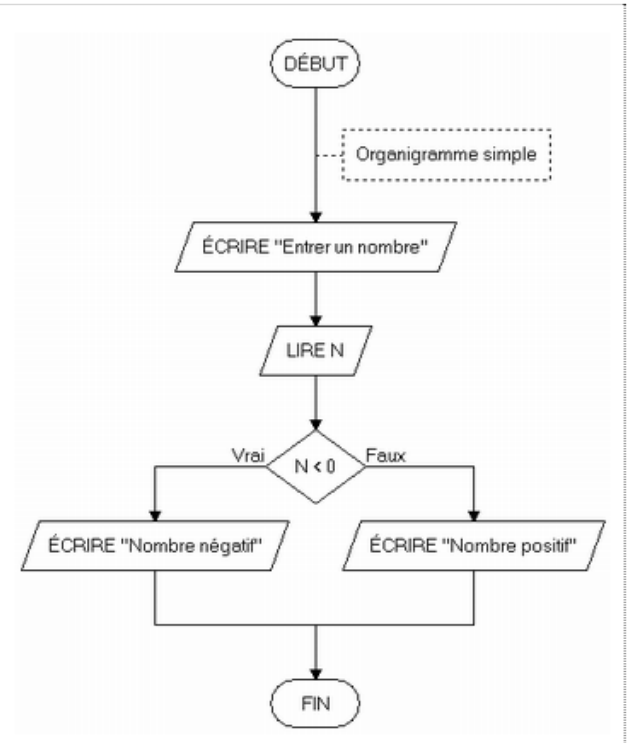



*Figure 9 : pseudo-code et organigramme de LARP*

[3] LARP est un logiciel éducatif s'appuyant sur un langage de programmation permettant le prototypage rapide d'algorithmes. **(Téléchargeable à partir de l'URL suivant :** https://fr.freedownloadmanager.org/Windows-PC/LARP.html**).**

LARP met à la disposition de l'utilisateur un espace de production d'algorithmes facile et convivial permettant au débutant de se familiariser rapidement avec le logiciel. L'utilisateur dispose donc d'un maximum de temps et de concentration pour programmer des algorithmes plutôt qu'à se familiariser avec l'interface ou avec une syntaxe de programmation aride comme le montre la capture d'écran de l'interface de LARP ci-dessous.

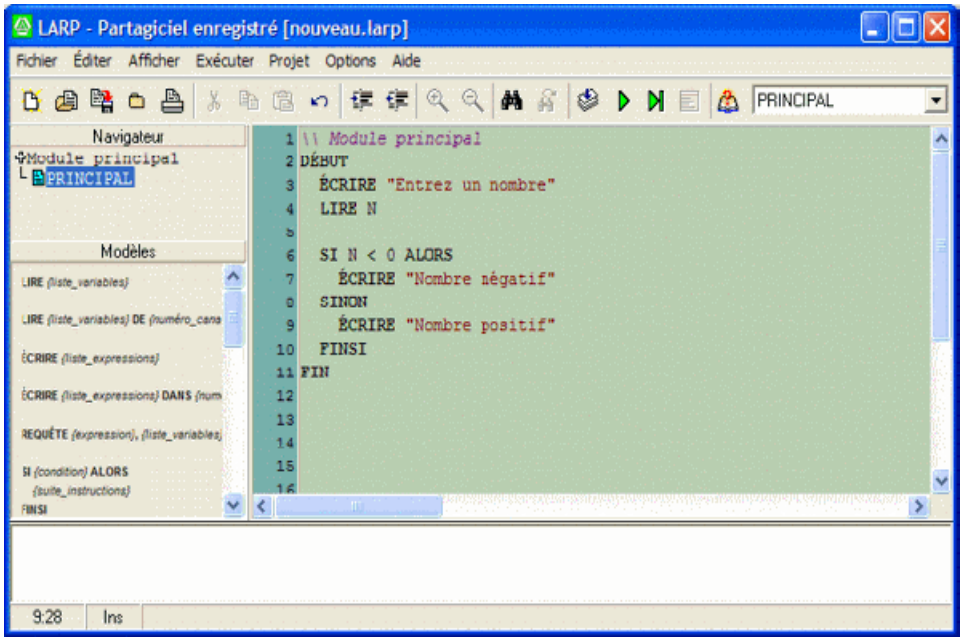



*Figure 10 : pseudo-code dans l'interface de LARP*

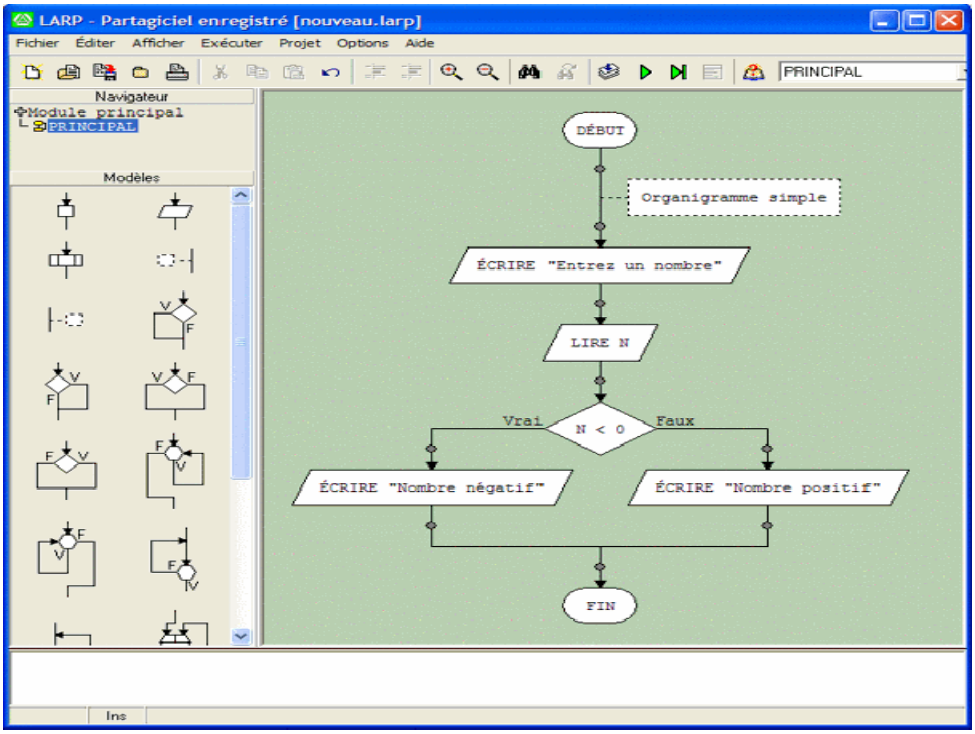



*Figure 11 : organigramme dans l'interface de LARP*

La souplesse du langage de programmation LARP ainsi que l'attrait de l'environnement de développement rend le logiciel particulièrement indiqué à l'enseignement de la programmation. L'enseignant peut se servir des pseudo-codes et/ou des organigrammes comme illustré à la figure 11 ci-dessus pour expliquer les concepts de programmation (les conditions, les boucles et la modularité). En pratique, les étudiants peuvent implémenter puis simuler à l'aide de LARP les algorithmes présentés par l'enseignant.

Dans le but de rendre son exploitation accommodante en milieu éducatif, LARP dispose d'une aide contextuelle qui présente sa syntaxe sous une forme pédagogique. Ainsi, l'aide en ligne permet à l'utilisateur d'apprendre la syntaxe de LARP et d'apprendre en même temps à exploiter des concepts importants en programmation à l'instar des variables, des structures conditionnelles et répétitives, etc. Ces concepts sont expliqués et suivis d'exemples qui facilitent leur compréhension.

De plus, l'espace de programmation de LARP possède un débogueur intégré qui lui permet d'exécuter un algorithme pas à pas tout en contrôlant l'évolution du contenu de ses variables. Le débogueur de LARP peut même nous permettre de voir sous le forme d'une animation l'exécution des différentes instructions d'un algorithme.

### 2.2.3.4. PluriAlgo[4]

PluriAlgo est un logiciel qui utilise plusieurs langages (pluri-langages) d'initiation à l'algorithmique, pour le lycée ou le premier cycle universitaire. Cette application souhaite se conformer aux différentes pratiques en lycée (Mathématiques, Technologie) ou en premier cycle universitaire. Il souhaite également simplifier la programmation dans plusieurs langages algorithmiques (AlgoBox, Javascool...), professionnels (Javascript, Python...) ou mathématiques (Xcas, Scilab...). PluriAlgo est présent sous deux versions : une en ligne qui ne nécessite aucune installation, et une sur poste fixe.

L'application propose des outils adaptés aux principales notions algorithmiques. Ces outils sont intégrés à l'intérieur de deux fenêtres pour la version en ligne. Trois onglets à droite de la fenêtre donnent la possibilité de rédiger ou de traduire des programmes dans dix autres langages de programmation:

- l'onglet « Principal » est en charge de la gestion des variables et de l'introduction de sous-programmes.
- l'onglet « Si » sert à écrire les instructions conditionnelles.
- l'onglet « Boucles » quant à lui permet l'introduction de boucles et des opérations de sommation, comptages, etc.

---

**(4)** PluriAlgo est un logiciel cherchant à s'adapter à la diversité des pratiques dans l'enseignement de l'algorithmique en lycée (ou en premier cycle universitaire) en facilitant l'écriture d'algorithmes dans plusieurs langages pédagogiques et en facilitant le passage d'un langage à un autre grâce à un traducteur. **(Disponible en ligne à partir de l'URL suivant :** http://raffinat.perso.univ-pau.fr/plurialgo/gwt/index.html**).**

La fenêtre de gauche est composée de deux onglets d'édition (Editeur1, Editeur2), d'un onglet de documentation (Aide) et enfin d'un onglet (Exécution) qui permet d'introduire au sein de l'application des outils externes qui donnent la possibilité d'exécuter des programmes Python, AlgoBox, etc. En effet, cette application donne la possibilité grâce à ses outils de réutiliser sous une nouvelle forme l'existant:

- le mécanisme de « **reformulation** » sert à examiner d'une autre manière en utilisant par exemple des sous-programmes ou des enregistrements des problèmes précédemment résolus d'une manière élémentaire lorsque des notions plus évoluées n'avaient pas été enseignées.

- le mécanisme de **« traduction** » simplifie le passage d'un langage pédagogique vers un langage professionnel, par exemple le passage de Larp, Javascool, Python, AlgoBox... en Visual Basic, JavaScript, PHP..., ou éventuellement de traduire un langage pédagogique en un autre.

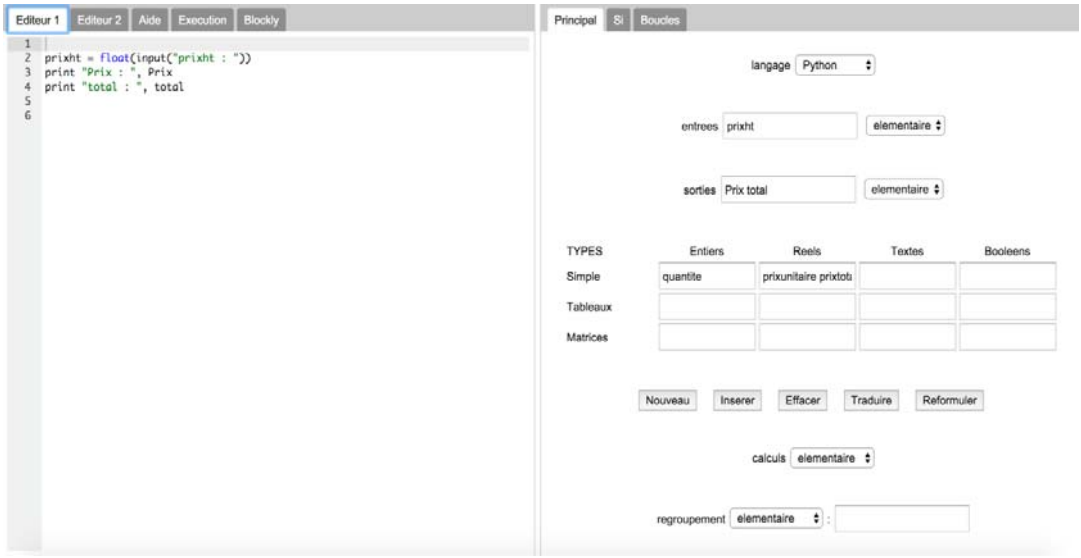



*Figure 12 : Interface de PluriAlgo*

PluriAlgo est une application qu'il n'est pas nécessaire d'installer. Il dispose d'une interface simple (figure 12) et d'une documentation fournie, il peut être utile pour les enseignants qui souhaitent débuter ou progresser en algorithmique. Cette application souhaite tenir compte des multiples pratiques dans l'enseignement de l'algorithmique en lycée et en premier cycle universitaire en facilitant :

- L'écriture d'algorithmes dans différents langages de programmation pédagogiques.

- Le passage d'un langage vers un autre grâce à un traducteur intégré.

En définitive, PluriAlgo peut servir soit pour faire une initiation à l'algorithmique, soit pour réaliser un approfondissement en algorithmique à l'aide d'outils comme les sous-programmes et les structures de données. Il simplifie la résolution de nombreux

exemples pédagogiques. L'apport de cette application dans un processus enseignement-apprentissage dépend non seulement du problème traité, mais aussi du langage de programmation qui sera étudié par les apprenants à la suite de l'algorithmique car certains langages de programmation sont plus adaptés à être utilisés à la suite de PluriAlgo.

### 2.2.3.5. Linotte[5]

Au terme de cette présentation de quelques logiciels destinés à la simulation des algorithmes et avant la présentation du logiciel support de notre recherche, nous allons présenter un logiciel libre qui est dédié à l'usage qui nous préoccupe.

Linotte est un langage de programmation qui possède la singularité d'offrir aux développeurs une plateforme de travail qui utilise une syntaxe en Français. De ce fait, son apprentissage est rapide. Il s'agit d'un langage libre, dont l'apparition remonte à l'année 2005 et qui permet aux novices d'apprendre à programmer facilement et de manière conviviale. Il offre la possibilité à chaque instant d'interroger les fondamentaux de la programmation : l'algorithmique. En orientant uniquement son attention sur l'aspect de la réflexion, aidé en cela par une syntaxe claire et facile, on apprendra non pas un langage mais plutôt la démarche intellectuelle que requiert la création d'un programme. Linotte est un langage multi-paradigmes : on peut en effet commencer par adopter une logique de programmation impérative, ensuite évoluer vers la programmation fonctionnelle et celle orientée objet. On pourra ainsi, à l'aide de cette application faire ses premiers pas dans la programmation avant d'évoluer vers des univers plus complexe.

Inspiré des langages Basic, Logo et Java, Linotte est conçu pour faciliter la compréhension des mécanismes et des logiques de la programmation dans la mesure où il ne s'appuie pas sur les notions de mathématique ou de technique. Son environnement de développement entièrement en français simplifie également son apprentissage. C'est une application qui est structurée et fortement typée (seulement 5 types de base), elle utilise un langage objet, dopé à la programmation orientée prototype. Ce logiciel est capable d'implémenter des algorithmes récursifs (notion de fonctions, de paramètres et variables locales), il est extensible par l'ajout de greffons écrits en langage Linotte, Java, Ruby ou en Python. Comme tous les logiciels libres, Linotte est en perpétuelle évolution.

Linotte fonction sous Mac, Linux et sous Windows.

---

**(5)** Inspiré des langages Basic, Logo et Java, Linotte permet d'assimiler rapidement les mécanismes et les logiques de la programmation car il ne requiert aucune notion mathématique ou technique. **(Téléchargeable à partir du lien suivant :** http://www.toucharger.com/fiches/windows/linotte/33459.htm**).**


*Figure 13 : La fenêtre principale de Linotte*

La fenêtre principale de Linotte présentée dans la figure ci-dessus est composée de neuf (9) parties :

1- Le cahier, l'élément principal dans lequel sera écrit et modifié le programme (le livre) ;

2- Le sommaire. Il permet de visualiser l'organisation de votre livre.

3- Le tableau. Il affiche les résultats du livre, ainsi que les messages d'erreur.

4- L'espace de travail. C'est l'endroit où l'on crée un nouveau livre dans un dossier prédéfini ou un nouveau répertoire.

5- Le tutoriel. Il permet d'accéder à tous les exemples de programmation en Linotte disponibles.

6- Les favoris. La mise en favoris d'une ligne permet de pouvoir accéder directement à cette ligne.

7- La boite à espèces. Elle permet d'afficher toutes les caractéristiques et fonctions du vocabulaire utilisé en Linotte.

8- L'éditeur de greffons. Les greffons ont pour but d'enrichir le Linotte à partir de programmes écrits dans d'autres langages.

9- La barre d'outils. Elle se compose de boutons et menus.

Afin d'écrire un livre (un programme), Linotte utilise trois principaux outils : les fonctions, les variables et les verbes. On peut également introduire des commentaires à la suite des lignes de programme. L'utilisation de ces outils spécifiques se fait parallèlement avec des éléments de programmations plus conventionnels pour certains :

les conditions, les boucles, mais d'autre moins connus mais dont on peut avoir des équivalents dans les autres logiciels : les casiers, les drapeaux et les espèces.

Linotte permet également de faire de la programmation graphique en donnant la possibilité d'accéder à des espèces dites graphiques. Les espèces graphiques sont des objets (texte, figures géométriques, etc.) que l'ont va afficher sur la toile sur la base de l'attribut et des coordonnées. La toile étant le support qui donne la possibilité de mettre ces objets à l'écran. Les espèces étant prédéfinies par l'interpréteur, on ne doit donc plus les créer. Ces espèces sont :

- le graffiti ;
- le parchemin ;
- les figures géométriques ;
- le graphique et le patron ;
- le praxinoscope ;
- le scribe.

Enfin, Linotte donne la possibilité d'utiliser les outils qui ont été développés dans un atelier linotte sur d'autre ordinateur ne disposant pas de cette application. Pour cela, il est important de créer des interfaces homme-machine (IHM) aussi appelé GUI, ce qui signifie « Graphic User Interface ». Les IHM sont tout simplement les interfaces graphiques avec lesquelles nous communiquons lorsque nous utilisons une application quelconque. Elles sont composées de fenêtres, de boutons, de menus, de cases à cocher, etc. Pour la programmation d'IHM, Linotte utilise plusieurs outils comme :

- les formulaires ;
- les popups ;
- insérer une zone de texte ;
- proposer un choix à l'utilisateur ;
- la table ;
- le menu ;
- les onglets ;
- l'indicateur ;
- l'espèce xtoile ;
- le scroller ;
- le slider.

## 2.2.3.6. AlgoBox[6]

Très utilisé dans l'enseignement des mathématiques en séries scientifiques dans les lycées en France, AlgoBox est selon ses développeurs un logiciel d'initiation à l'algorithmique. Au lancement, on observe de façon distincte deux zones de texte : « présentation de l'algorithme » et « code de l'algorithme » (figure 14). La première zone est un en-tête qui permet de présenter l'algorithme, tandis que la seconde permet de saisir le pseudo-code de la déclaration des variables et du corps de l'algorithme. Une série de boutons et d'onglets permettent l'insertion, la modification et la suppression d'objets.

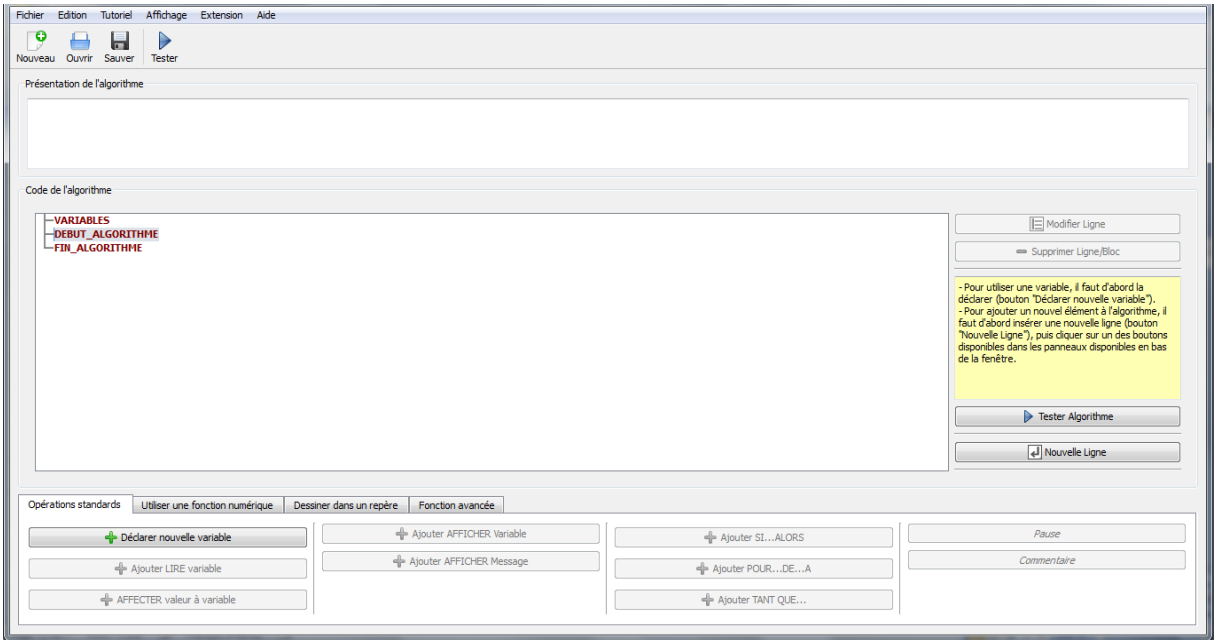



*Figure 14 : écran d'accueil d'AlgoBox*

Après le lancement de l'interface d'accueil, les instructions de l'algorithme sont construites à partir d'un mini-langage algorithmique de type « pseudo-code ». À l'exception des fonctions mathématiques, toutes les instructions sont en français. Des instructions de base qui sont introduites par des clics sur les boutons de l'interface servent à construire progressivement et de façon hiérarchique et structurée l'algorithme. Ceci permet à l'utilisateur de se concentrer sur l'élaboration de l'algorithme et ainsi s'affranchir des difficultés liées à son formalisme (figure 15).

[6] **AlgoBox est un logiciel libre** et gratuit d'aide à l'**élaboration et à l'exécution d'algorithmes. Doté d'une interface en français, claire et ergonomique, il est d'une prise en main facile et rapide. (Téléchargeable à partir de l'URL suivant :** http://www.xm1math.net/algobox/download.html**).**

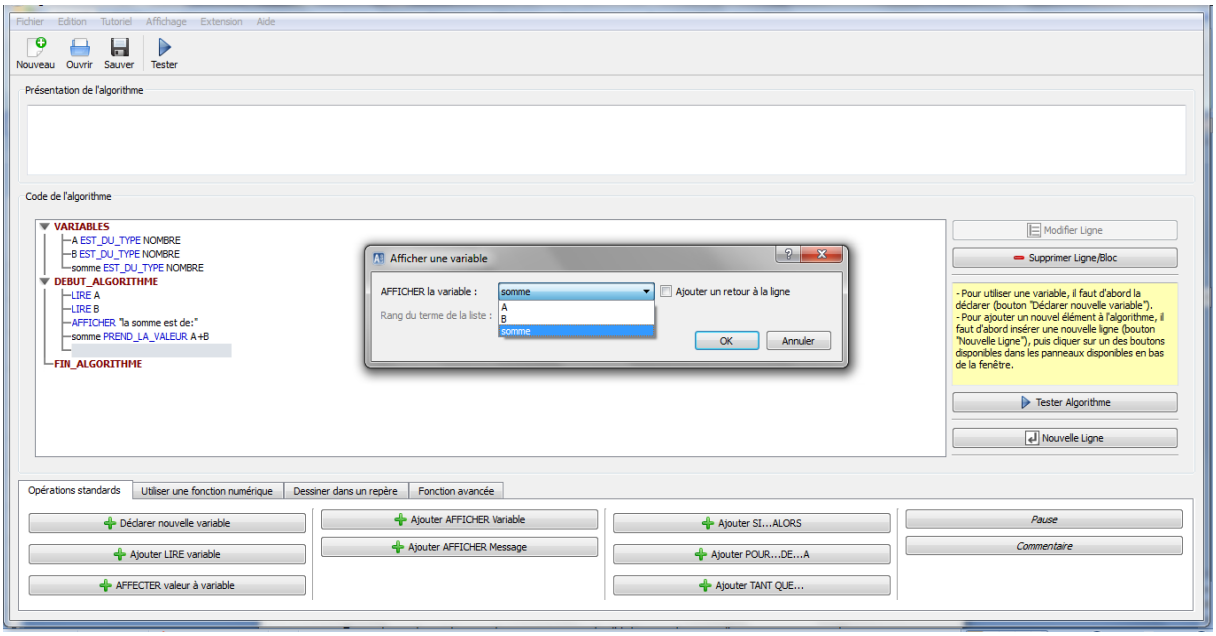



*Figure 15 : affichage d'une variable*

Lors de l'édition d'algorithmes plus complexes, un mode « éditeur de texte » permet aux utilisateurs avancés de taper directement les instructions de l'algorithme à l'aide d'un éditeur incorporé muni de certaines fonctions permettant de faciliter la saisie et la compréhension de l'algorithme.

Lorsqu'on a terminé d'éditer l'algorithme, un bouton « Lancer Algorithme » (figure 16) donne la possibilité d'exécuter l'algorithme. L'interface de travail d'AlgoBox présente également un bouton « Tester Algorithme » (figure 15) qui permet l'exécution de l'algorithme ligne après ligne afin d'observer l'exécution de chaque ligne du programme. Cette exécution s'appelle le mode « pas à pas pédagogique ».

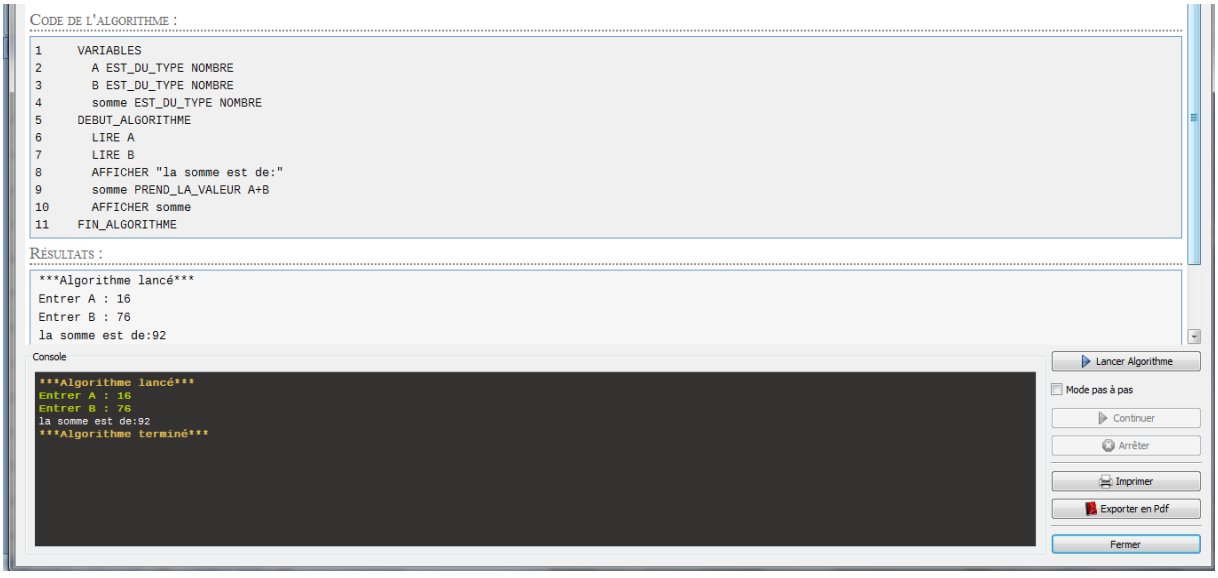



*Figure 16 : fenêtre d'exécution d'Algobox*

Au regard de la simplicité d'utilisation de cette application, AlgoBox dispenserait effectivement l'apprenant d'une grande partie des tâches liées à l'apprentissage du formalisme syntaxique de l'algorithmique. Il offre, en effet, à l'apprenant la possibilité de rentrer le pseudo-code soit en cliquant sur des boutons, soit par le biais de propositions prédéfinies.

En revanche, l'utilisation de ces propositions de pseudo-code nécessite cependant la connaissance des structures d'un algorithme, des fonctions prédéfinies et d'un vocabulaire spécialisé même si celui-ci ne présente aucune barrière linguistique car étant en français.

Rappelons que pour un enseignement d'initiation à l'algorithmique, certaines compétences doivent être visées par l'enseignement et ce sont ces compétences qui déterminent le choix d'une application qui servira de support afin d'accompagner l'enseignant pour que les apprenants acquièrent ces compétences. Nous pouvons citer :

- L'écriture complète d'un algorithme.
- La compréhension et l'analyser un algorithme préexistant.
- La modification d'un algorithme dans le but d'obtenir un résultat précis.
- L'analyse de la situation, c'est-à-dire identifier les données d'entrée, de sortie, le traitement...
- La mise au point d'une solution algorithmique, c'est-à-dire la méthode d'écriture d'un algorithme en langage courant avec respect du code, ainsi que l'identification des structures répétitives et alternatives, des opérations de lecture, d'affichage...
- La validation de la solution algorithmique par des traces d'exécution et des jeux d'essais simples.
- L'adaptation de l'algorithme aux exigences du langage de programmation, c'est-à-dire l'identification si cela est utile de la nature des variables...
- La validation d'un programme simple.

Dans le cas de l'ENSET de Libreville, il nous fallait trouver une petite application simple à utiliser, qui ne nécessitait pas une formation particulière tant pour les enseignants que pour les apprenants, tout en visant la plupart des compétences citées plus haut. Il nous fallait éviter que les apprenants soient confrontés à des difficultés supplémentaires. En effet, la première année à l'ENSET est un tronc commun dans lequel sont inscrits des élèves professeurs de spécialités différentes. L'ENSET ne forme pas des programmeurs, ni des professeurs de programmation ou d'informatique, l'enseignement de l'algorithmique et de la programmation est mis au service d'activités de résolution de problèmes pour les sciences, c'est également un socle d'apprentissage de la logique en générale et de la logique programmée en particulier.

C'est ainsi que contrairement aux autres applications citées ci-dessus, AlgoBox nous a semblé plus apte à répondre à nos attentes. Au niveau de la syntaxe, AlgoBox offre une programmation en pseudo-code avoisinant l'écriture d'un algorithme sur papier, ce qui nous dispense des difficultés liées à l'apprentissage du formalisme de la syntaxe d'un langage de programmation. L'exécution en mode pas à pas offre des perspectives pédagogiques intéressantes dans la mesure où la valeur de chaque variable est affichée au fur et à mesure du déroulement de l'algorithme. Enfin, AlgoBox permet donc d'être en conformité avec les objectifs et les exigences en première année à l'ENSET de Libreville afin de tenter de booster les résultats dans l'enseignement de l'algorithmique sans rebuter la majorité des élèves qui ont des difficultés dans les enseignements scientifiques, et sans pénaliser ceux qui n'en ont pas. Le fait d'être très utilisé en France a aussi beaucoup œuvré en faveur d'AlgoBox.

AlgoBox servira donc de plateforme que nos sujets utiliseront au cours de la phase expérimentale de notre recherche. En effet, le dispositif expérimental que nous avons choisi nous oblige, afin de vérifier nos hypothèses à faire travailler un groupe de sujets (le groupe dit expérimental) avec cette application de simulation d'algorithmes entre le pré-test et le post-test au cours de plusieurs séances de travaux pratiques. Quelles sont alors les hypothèses qui seront mis à l'épreuve dans ce dispositif expérimental ?

## 2.3. Questions et hypothèses de recherche

Au regard de notre cadre théorique, nos questions et hypothèses de recherche vont avoir pour point d'ancrage la théorie de la charge cognitive notamment son aspect lié à la réduction de charge cognitive intrinsèque qui aurait pour effet d'améliorer l'apprentissage. Les types de savoirs mise en œuvre, en l'occurrence la prise en compte dans l'accomplissement d'une tâche des savoirs implicites portés par la tâche doit contribuer ainsi à faciliter l'apprentissage. La réalisation de la tâche, l'écriture d'un algorithme, s'appuie sur une démarche de résolution de problèmes, l'apprentissage, la mise en œuvre et le respect des règles du formalisme. Et enfin, l'usage d'un simulateur dans l'apprentissage de l'algorithmique prend en compte une démarche active qui permet à l'apprenant d'avoir une aide importante qui le soulage des contraintes liées au formalisme et qui rend la ressource cognitive disponible afin de résoudre le problème présent dans la tâche prescrite en algorithmique.

Au regard de ces éléments, nous envisageons deux questions de recherche.

Pour la première question, nous allons nous demander s'il est possible de faciliter l'apprentissage de l'algorithmique en introduisant dans le processus enseignement-apprentissage le logiciel d'aide à l'élaboration d'algorithme « AlgoBox ».

La seconde interrogation va nous amener à nous demander si, au niveau didactique, la prise en compte, par l'enseignant, des savoirs implicites liés à la tâche prescrite, permet à l'étudiant de mieux mobiliser les connaissances adéquates afin de faciliter la compréhension de la situation-problème.

De la première question de recherche découle l'hypothèse selon laquelle en s'appuyant sur les travaux relatifs à la charge cognitive, l'utilisation d'un outil comme AlgoBox devrait réduire la charge cognitive et de ce fait faciliter l'apprentissage de l'algorithmique en soulageant en partie l'apprenant de l'activité liée à l'apprentissage du formalisme par réduction de la charge cognitive intrinsèque (Paas, 1994; Sweller, 2010) ».

Concernant la deuxième question, nous retenons que l'élaboration d'un algorithme sollicite chez un apprenant l'accomplissement de deux tâches, l'une est la résolution du problème et l'autre est la gestion du formalisme de l'algorithme. Ainsi, la réactivation des savoirs implicites portés par la tâche prescrite, et qui seront sollicités dans l'élaboration de la solution, devrait contribuer à faciliter la compréhension et la résolution du problème, tandis qu'AlgoBox va soulager l'apprenant dans la gestion du formalisme de l'algorithme.

Ainsi, après avoir identifié certains indicateurs dont la maitrise témoigne de la compréhension et de la mise en œuvre des savoirs appris en cours, nous réaliserons une première expérimentation. L'analyse des résultats de cette première expérimentation nous permettra de confirmer ou d'infirmer les difficultés des apprenants, puis d'identifier les savoirs qui leurs posent des problèmes grâce à une analyse de la tâche prescrite, plus précisément de leur production. Cette analyse des productions des apprenants conformément à chaque indicateur nous renseignera sur la maitrise ou non de l'indicateur (savoir-faire pour l'apprenant). Dans une seconde expérimentation, nous allons introduire dans le processus enseignement-apprentissage de l'algorithmique une application d'aide à l'élaboration d'algorithmes. Cette application permettra aux apprenants, en travaux pratiques, après l'écriture d'une proposition d'algorithme en travaux dirigés, de simuler leur solution, de la réajuster si elle ne marche pas, puis, *in fine*, d'arriver avec l'aide, si nécessaire de l'enseignant à la solution fonctionnelle. Au terme de ce processus, les données recueillies seront analysées et interprétées conformément à nos hypothèses de recherche.

# 3. Expérimentations

## 3.1. Présentation de la première expérimentation

### 3.1.1. Objectif visé

Cette première expérimentation a pour but d'observer les sujets dans l'accomplissement d'une tâche prescrite, c'est-à-dire dans la résolution d'une tâche d'algorithmique (écrire un algorithme) comme cela se fait actuellement à l'ENSET de Libreville, donc sans l'utilisation d'un logiciel d'aide à l'élaboration d'algorithmes.

L'observation devra donc mettre en évidence, dans un premier temps, la double difficulté des sujets à comprendre la tâche prescrite et à élaborer l'algorithme qui leur est demandé ; puis l'analyse de la production des étudiants nous permettra d'identifier certains indicateurs à partir desquels on focalisera notre étude. En effet, ces indicateurs correspondent à des savoirs qui seront sollicités dans le test. Au cours du test, l'apprenant aura à écrire des algorithmes en s'appuyant sur les connaissances adéquates. L'indicateur permettra alors de déterminer si les connaissances mobilisées par l'apprenant sont celles attendues ou pas. De plus, une analyse de ces indicateurs permettra de comprendre les difficultés rencontrées par les apprenants, les erreurs que ces derniers commettent lors de la restitution du savoir et du savoir-faire sous la forme de connaissances. Finalement, c'est l'analyse de l'activité des sujets au travers des différents indicateurs liés aux savoirs et savoir-faire portés par la tâche, qui nous permettra dans la première expérimentation de comprendre les difficultés liées à la réalisation de la tâche et d'essayer d'en déterminer les causes.

Dans la seconde expérimentation, ce sont les mêmes indicateurs qui seront utilisés pour analyser les traces de l'activité des étudiants pour voir l'apport de notre application dans le processus enseignement-apprentissage.

Pour parvenir à résoudre les problèmes qui leur sont proposés, il est indispensable que les sujets comprennent les énoncés qui leur seront proposés. Comprendre l'énoncé c'est savoir ce qu'on leurs demande de faire (écrire un algorithme, c'est savoir ce qu'est un algorithme, connaitre ces différentes parties syntaxiques). C'est aussi être capable d'élaborer le but de la tâche prescrite et de faire le lien avec ce qu'est un algorithme (je peux très bien savoir poser le problème mais ne pas savoir le traduire en algorithme ; de même, je peux très bien savoir traduire des résolutions de problème arithmétique en algorithme, mais je ne sais pas résoudre le problème lui-même…), et produire des écrits qui respectent les caractéristiques d'un algorithme (structures sémantiques du langage). De plus, élaborer un algorithme c'est être capable de faire ce que l'énoncé leur demande : c'est-a-dire produire des écrits (codes) qui respectent les caractéristiques de

la structure sémantique d'un algorithme et qui en plus donnent une réponse au problème posé. On est donc capable de le faire quand on peut résoudre le problème qui est posé puis d'établir la structure d'un algorithme (par exemple connaitre la formule mathématique dont on a besoin afin d'en extraire les variables de la partie déclarative de l'algorithme). On est capable de le faire aussi quand on connait la syntaxe associée à la réponse préconisée. En définitive, l'apprenant doit, pendant et après la lecture de l'énoncé, pouvoir mobiliser un ensemble de connaissances liées aux savoirs sollicités, relatifs au cours (savoirs « explicites ») ou pas (savoirs « implicites »), afin d'élaborer l'algorithme relatif au problème posé par la tâche prescrite.

Lorsque nous lisons un énoncé, nous pouvons parfaitement comprendre ce qu'on nous demande de réaliser mais ne pas être en mesure de le faire, soit à cause d'un oubli relatif à une formule par exemple, soit par oubli d'une règle syntaxique. La lecture d'un énoncé doit donc permettre à l'apprenant de produire une solution qui est l'élaboration d'un algorithme. Quand il lit l'énoncé, l'apprenant est confronté à une double activité cognitive (charge cognitive), c'est-à-dire qu'il va dans un premier temps déployer de la ressource cognitive pour comprendre le problème, donc le résoudre, puis il va utiliser d'autre ressources cognitives pour identifier, choisir et utiliser les éléments syntaxiques nécessaires pour écrire l'algorithme. Pour faciliter cette activité cognitive, l'utilisation d'un vocabulaire spécifique dans l'énoncé va déclencher chez l'apprenant des connaissances propres à la situation problème pour la compréhension de l'énoncé. Donc si les connaissances déclenchées sont effectivement utilisables par la suite pour l'élaboration de l'algorithme alors l'apprenant sait ce qu'on lui demande de faire.

Enfin, au cours de cette première expérimentation, nous avons également effectué une observation des sujets pendant l'élaboration de la tâche prescrite (pendant qu'ils passaient le test). Cette observation indirecte non armée par le biais d'une grille d'observation élaborée à partir d'items spécifiques, adaptés à notre usage nous a permis de faire une analyse comportementale des sujets. Ainsi, en fonction des items, les sujets auront un comportement d'inhibition, d'introversion, dans un cas, ou d'extraversion s'appuyant sur des observations d'excitabilité ou/et d'instabilité. Pour cette expérimentation, la grille de recueil des données était constituée de deux parties. La première partie intitulée Introversion/inhibition était composée de quinze (15) items tandis que la seconde partie comprenait également quinze (15) items et était intitulée Excitabilité/instabilité/extraversion. Nous avons effectué cinquante-six (56) observations des sujets au cours de l'exécution de la tâche prescrite à la première expérimentation (dix-sept sujets) et au post test de l'expérimentation finale (trente-neuf sujets). En définitive, nous avons observé tous les sujets qui ont travaillés avec nous au cours de nos différentes expérimentations. Il s'agissait pour nous, assisté de quelques jeunes enseignants d'évaluer selon des items de notre grille sur une échelle de grandeurs allant de un (1) à cinq (5) le comportement du sujet au cours de l'élaboration de la tâche prescrite (voir grilles d'observation en annexe 7 & 9).

## 3.1.2. Échantillonnage

Pour réaliser cette première expérimentation, nous avons travaillé avec les élèves-professeurs de première année (Licence 1), sciences et techniques industrielles (STI) de l'École Normale Supérieure de l'Enseignement Technique (ENSET) de Libreville. Cette institution a entre autre mission, la formation des enseignants des disciplines technologiques et professionnelles pour les lycées techniques ou professionnels. Il s'agit de dix-sept (17) admis à un examen national qui ont constitué l'ossature de notre échantillon pour cette première expérimentation.

Le tableau 1 ci-dessous présente les caractéristiques de notre échantillon :

| | Type de baccalauréat | | | | | | Avoir déjà fait de l'algorithmique | | Sexe | | TOTAL |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | SM | MI | F1D | F1 | F3 | F4 | Oui | Non | M | F | |
| Échantillon | 3 | 1 | 1 | 6 | 5 | 1 | 1 | 16 | 13 | 4 | 17 |

*Tableau 1 : caractéristiques de l'échantillon*

## 3.1.3. Instrument d'investigation

Dans le cadre de notre observation, nous avons identifié pour l'ensemble du cours trente-trois (33) savoirs explicites. Pour éprouver la majorité de ces savoirs, nous nous sommes appuyés sur quatre exercices convoquant quatre principaux savoirs implicites couvrant quelques notions du parcours scolaire de l'enseignement secondaire :

- Les nombres pairs et les nombres impairs : sixième (6e);
- Le PGCD (Plus Grand Commun Diviseur) : cinquième (5e), quatrième (4e);
- Le Carré d'un nombre : troisième (3e);
- La résolution d'une équation du second degré : second cycle (lycée).

Ce choix de savoirs de l'enseignement secondaire est dicté par le fait que nos apprenants entrent dans l'enseignement supérieur, de ce fait, durant leur parcours scolaire dans l'enseignement secondaire, ils ont étudié les quatre notions sollicitées par notre test. Ces notions apparaissent dans le programme officiel et les manuels de mathématique en usage en république gabonaise. Ainsi, une trentaine de savoirs et savoir-faire explicites ont été testées. Les quatre principaux savoirs implicites ont généré quinze (15) savoirs et savoir-faire.

Lorsque nous analysons les quatre tests que nous préconisons, pour chacun d'eux, nous mettons en évidence deux types de savoirs dont doivent disposer les apprenants pour être capable de traiter ces exercices : des savoirs explicites (qui découlent directement des notions enseignées en cours), et des savoirs implicites (qui eux n'ont pas été directement étudié lors de l'enseignement mais dont la maitrise est nécessaire pour résoudre les exercices posés). Chacun de ces savoirs explicites et implicites se

mobilisant un ensemble de connaissances procédurales (correspondant à des savoir-faire).

Pour réaliser notre analyse, c'est-à-dire identifier les points sur lesquels les étudiants ont des difficultés, nous avons effectué une analyse de la tâche. Cette analyse visait pour objectif d'identifier des savoirs et savoir-faire en jeu dans la tâche prescrite.

Cette première expérimentation s'est déroulée après la clôture du module consacré à l'algorithmique, la semaine qui précédait celle consacrée aux révisions et deux semaines avant l'examen de fin de semestre. Les étudiants avaient donc déjà suivi tout l'enseignement consacré à l'algorithmique ; ils avaient déjà fait des travaux dirigés avec l'enseignant et aussi des devoirs sur table sur différentes parties du cours. Ils rentraient donc dans la semaine de révision qui précède l'examen de fin de semestre. Nous avons pour les besoins de l'expérimentation convoquée tous les étudiants de première année, dix-sept ont répondu positivement à notre invitation.

Le test était constitué de quatre (4) exercices que les étudiants devaient résoudre en deux heures soit en moyenne trente minutes par exercice. Le test étant libre, la consigne leur avait été donnée qu'ils avaient le droit de consulter toutes les sources d'informations dont ils disposaient : ils pouvaient regarder les notes de cours, discuter entre pairs, écrire au brouillon tout ce qui leur passait à l'esprit pendant qu'ils réfléchissaient sur le sujet, les déplacements dans la salle et à l'extérieur étaient aussi autorisés…

La construction de notre test a eu comme point de départ le contenu du cours d'algorithmique. L'analyse de ce contenu dans son état actuel nous a permis de mettre en évidence une trentaine de savoirs (33) que l'élève doit apprendre. Pour être capable de résoudre un problème (traiter un exercice), les connaissances liées à ces savoirs doivent être traduits en connaissances procédurales liées aux savoir-faire. Ne pouvant pas soumettre à l'expérimentation tous les savoirs faute de temps et moyens, car il aurait fallu dans ce cas faire beaucoup plus d'exercices et donner plus de temps aux étudiants, nous avons identifié parmi la trentaine de savoirs, ceux qui sont nécessaires à l'apprenant pour qu'il soit en mesure d'écrire un algorithme élémentaire. Pour ce faire, nous avons identifié vingt-neuf (29) savoir-faire explicites associés au quinze (15) savoir-faire implicites convoqués par nos quatre exercices qui sont inventoriés dans le tableau 2 ci-dessous :

| Savoir-faire explicite | Savoir-faire implicite |
|---|---|
| Affectation d'une fonction | Calcul du PGCD |
| Affectation incrémentation | Calcul du discriminant |
| Affectation initialisation | Carré d'un nombre |
| Affectation d'une expression/valeur | Discussion sur le discriminant |
| Afficher plus d'une variable | Équation du premier degré |
| Afficher texte et variable | Équation du second degré |
| Afficher un texte | Fonction carrée |
| Afficher une variable | Fonction racine carrée |
| Alternative double | Forme de la racine double |
| Alternative multiple | Forme des deux racines |
| Alternative simple | Nombre impair |
| Déclaration de plus d'une variable | Nombre pair |
| Déclaration d'une constante | Notion de PGCD |
| Déclaration d'une variable | Résolution d'une équation du 2nd degré |
| Fonction prédéfinie | Reste d'une division d'un entier par 2 |
| Imbrication d'alternatives | |
| Imbrication alternative/itérative | |
| Itération avec test à l'entrée | |
| Itération avec test à la sortie | |
| Itération sans test | |
| Lecture de plus d'une variable | |
| Lecture d'une variable | |
| Opérateurs arithmétiques | |
| Opérateurs logiques | |
| Opérateur modulo | |
| Opérateurs relationnels | |
| Structure du corps d'un algorithme | |
| Structure d'un algorithme | |
| Structure d'une déclarative | |

*Tableau 2 : liste de savoir-faire explicite et implicite*

Il nous parait important de rappeler qu'un savoir explicite ou implicite peut être su par l'apprenant, c'est-à-dire que l'apprenant sait qu'il va utiliser dans la résolution de l'exercice tel ou tel savoir implicite ou explicite, mais ne pouvoir le traduire en ligne d'algorithme syntaxiquement correcte. C'est ainsi qu'afin de mettre en évidence les difficultés des apprenants, c'est-à-dire analyser les savoir-faire incriminés (non réussis) au cours d'une analyse des productions des apprenants pendant notre observation.

Nos exercices ont été élaborés de telle sorte que les étudiants y rencontrent deux niveaux de difficultés. Le premier niveau se compose s'exercices simples car sollicitant des savoirs et savoir-faire plus courant, souvent utilisé, donc plus familiers. Quant au

second niveau, il fait appel à des savoirs et savoir-faire plus complexes (voir annexe 12 : énoncés et propositions de solutions).

## 3.1.4. Résultats de la première expérimentation

### 3.1.4.1. Analyse des traces écrites

Pour analyser les productions des apprenants, nous avons identifié un ensemble d'indicateurs qui vont nous renseigner sur la capacité des apprenants à mettre en œuvre une notion de cours sous la forme d'un algorithme. Cette mise en œuvre étant le fait que l'apprenant arrive à associer un savoir suscité par le test à un savoir étudié en cours, puis traduire correctement ce savoir en un savoir-faire.

Nous avons donc pour notre expérimentation fait de l'ensemble des vingt-neuf savoir-faire explicites et des quinze savoir-faire implicites des indicateurs de mise en œuvre des savoir-faire. Quand un indicateur est parfaitement mise en œuvre dans l'/les exercice (s) qui fait/font appel à lui, ont dira que le savoir est connu de l'apprenant. On peut donc déduire que si l'ensemble des indicateurs sont parfaitement mis en œuvre dans tous les exercices, alors on conclura que les savoirs sont connus des étudiants.
Les tableaux 3 et 4 ci-dessous montrent le taux de réussite dans chaque indicateur par l'ensemble des apprenants dans tous les exercices.

| Indicateurs explicites | Taux de réussite |
|---|---|
| Affectation d'une fonction | 21/51 |
| Affectation incrémentation | 10/34 |
| Affectation initialisation | 9/34 |
| Affectation d'une expression/valeur | 28/68 |
| Afficher plus d'une variable | 9/51 |
| Afficher texte et variable | 1/68 |
| Afficher un texte | 24/68 |
| Afficher une variable | 24/68 |
| Alternative double | 4/68 |
| Alternative multiple | 2/51 |
| Alternative simple | 1/68 |
| Déclaration de plus d'une variable | 27/68 |
| Déclaration d'une constante | 0/51 |
| Déclaration d'une variable | 15/68 |
| Fonction prédéfinie | 21/34 |
| Imbrication d'alternatives | 0/34 |
| Imbrication alternative/itérative | 0/34 |
| Itération avec test à l'entrée | 8/34 |
| Itération avec test à la sortie | 0/34 |
| Itération sans test | 0/34 |
| Lecture de plus d'une variable | 18/51 |
| Lecture d'une variable | 15/68 |
| Opérateurs arithmétiques | 27/51 |
| Opérateurs logiques | 1/51 |
| Opérateur modulo | 0/17 |
| Opérateurs relationnels | 33/68 |
| Structure du corps d'un algorithme | 47/68 |
| Structure d'un algorithme | 48/68 |
| Structure d'une déclarative | 41/68 |

*Tableau 3: taux de réussite pour chaque indicateur de savoir explicite*

| Indicateurs implicites | Taux de réussite |
|---|---|
| Calcul du PGCD | 0/17 |
| Calcul du discriminant | 11/17 |
| Carré d'un nombre | 8/34 |
| Discussion sur le discriminant | 4/17 |
| Équation du premier degré | 3/17 |
| Équation du second degré | 2/17 |
| Fonction carrée | 21/34 |
| Fonction racine carrée | 14/17 |
| Forme de la racine double | 9/17 |
| Forme des deux racines | 8/17 |
| Nombre impair | 0/17 |
| Nombre pair | 0/17 |
| Notion de PGCD | 0/17 |
| Résolution d'une équation du 2nd degré | 7/17 |
| Reste d'une division d'un entier par 2 | 0/17 |

*Tableau 4 : taux de réussite pour chaque indicateur de savoir implicite*

Lorsque nous analysons la production des apprenants, les résultats montrent que pour les quarante-quatre indicateurs (29 explicites + 15 implicites) portés par les quatre exercices proposés, neuf sont réussis, soit un taux d'échec de 35/44 comme le montre le graphique ci-dessous.

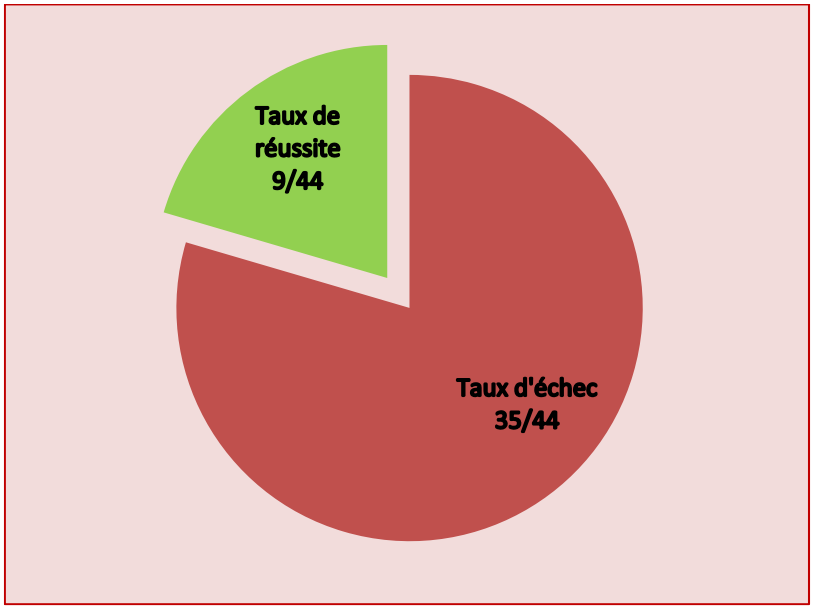



*Figure 17: Répartition des taux d'échec et de réussite pour l'ensemble*

Les indicateurs relatifs aux savoir-faire implicites sont le mieux réussis avec quatre sur quinze, soit un taux d'échec de 11/15, tandis que les indicateurs relatifs aux savoir-faire explicites sont le moins bien réussis avec cinq sur vingt-neuf, soit un taux d'échec de 24/29 comme indiqué dans les graphiques en figure 18 et 19 ci-dessous.

Ce résultat nous conduit au constat que se sont les savoirs explicites étudiés en cours que les apprenants ont le plus de difficultés à traduire en savoir-faire à mettre en œuvre dans les exercices.

Il apparait également que les indicateurs les moins biens réussis sont les indicateurs liés aux savoir-faire implicite. Ces dernières sont encore dans leur ensemble mieux réussi que les indicateurs explicites qui sont pourtant directement tirés du cours. Ce résultat est un peu surprenant mais peut ne pas être pertinente dans la mesure où en terme de quantité, il y a deux fois plus d'indicateurs explicites qu'implicites qui ont été testés.

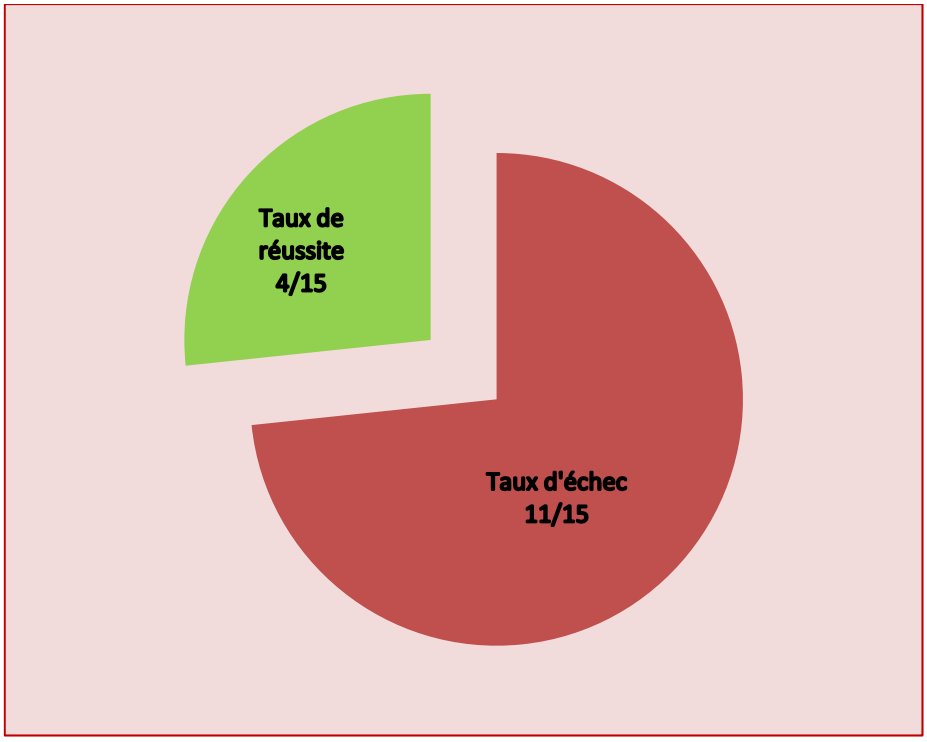



*Figure 18 : Répartition des taux d'échec et de réussite pour les savoir-faire implicites*

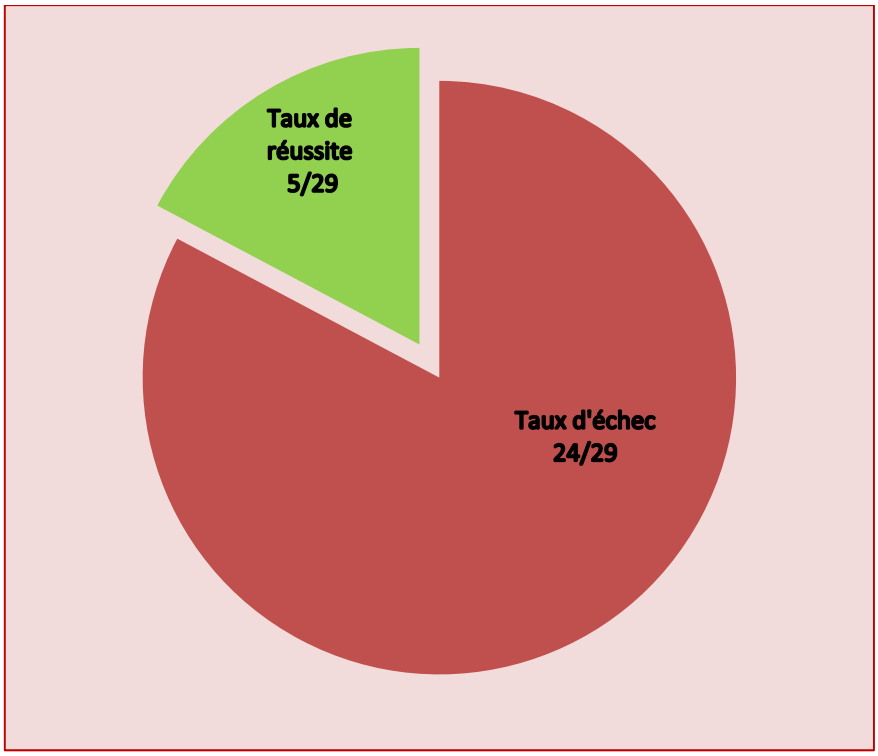



*Figure 19 : Répartition des taux d'échec et de réussite pour les savoir-faire explicites*

D'une manière générale, il se dégage aussi que lorsqu'un savoir implicite n'est pas connu de l'apprenant, ce dernier est incapable de produire la moindre solution. Ce constat nous l'avons sur l'exercice relatif au PGCD (Plus Grand Commun Diviseur). Dans cet exercice, presque tous les apprenants n'ont écrit aucune ligne de programme, nonobstant la notion de PGCD et le calcul du PGCD qui représentaient les savoir-faire implicites. Les autres procédures (savoir-faire) indispensables étaient aussi utilisées et plutôt réussies dans les autres exercices. En effet, l'examen de la réussite des indicateurs comme la structure d'un algorithme, la structure du corps d'un algorithme et la structure de la partie déclarative, qui sont très bien réussies par tous les apprenants dans tous les exercices, fait apparaitre que ce succès n'est qu'à 1/17 pour les deux premiers cités et à 3/17 pour le dernier.

### 2.1.4.2. Analyse des indicateurs non réussis

D'une manière générale, comme nous l'avons fait ressortir dans les tableaux 3 et 4, il y a non seulement des indicateurs qui nous ont permis de comprendre que certains savoir-faire ont été plutôt bien abordés par les apprenants, mais aussi que d'autres n'ont pas été réussis. Si la première situation ne nous enseigne pas dans la mesure où elle ne pose aucune question suscitant l'intérêt du chercheur, la seconde par contre mérite qu'on s'y intéresse un peu rien que pour se demander pourquoi ces savoir-faire n'ont – ils pas été réussis par les apprenants.

Concernant les savoirs implicites, ont peut dire que leur non réussite est liée justement à leur caractère implicite, c'est-à-dire ne faisant pas appel à des notions

apprises présentement mais plutôt antérieurement : c'est-à-dire ne correspondent pas aux savoirs que l'enseignant souhaite tester. Si l'élève n'as pas appris le savoir-faire, il ne peut pas « inventer » la connaissance procédurale qui, elle aussi, résulte d'un apprentissage. Et pour pouvoir utiliser cette connaissance procédurale, il faut interpréter correctement la situation-problème pour savoir si on peut l'utiliser ou non. Donc la non réussite des sujets sur les savoirs implicites est due semble t-il au fait qu'il n'ait pas construit la connaissance procédurale associée à ce savoir-faire ou éventuellement la situation-problème ne lui permet pas d'activer cette connaissance procédurale.

Concernant l'échec aux savoirs explicites, ce qui est dit pour les savoirs implicites reste valable du moins si le savoir-faire n'a pas été appris. Cependant, le caractère explicite du savoir soulève un doute. En effet, nous savons que le savoir-faire a été enseigné si l'on se confère au cours, mais l'apprenant ne construit pas la connaissance procédurale utilisable pour interpréter l'énoncé du problème, d'où l'échec. Ici on peut penser qu'il n'y a pas eu réellement apprentissage, et pourtant le savoir a été « enseigné ». Le processus enseignement-apprentissage présente donc des limites, il ne produit pas les résultats attendus. Un savoir-faire est « bien » enseigné lorsque le savoir-faire est appris par l'apprenant, c'est à dire l'apprenant a construit la connaissance procédurale utilisable pour résoudre la situation-problème. Le processus enseignement-apprentissage s'étant bien accompli, l'apprenant peut donc mettre en œuvre une solution au problème.

En effet, lorsque nous regardons les copies des apprenants et que nous analysons par rapport aux exercices proposés les indicateurs, nous constatons que pour qu'un exercice soit réussi, le savoir doit avoir été appris, quelque soit le cadre ou le contexte, ou doit avoir été enseigné. De plus, le savoir-faire associé doit aussi avoir été appris. Concrètement, un exercice est réussi quand l'élève a appris alors on peut dire qu'il a construit la connaissance procédurale associée à ce savoir ou savoir-faire. L'exercice est aussi réussi lorsque la situation-problème permet à l'apprenant d'activer la connaissance procédurale construite associée à ce savoir ou savoir-faire. En définitive, apprendre ici s'est soit construire la connaissance procédurale, soit activer la connaissance procédurale construite.

Nous identifions sur notre échantillon de quarante-quatre (44) indicateurs, trente-cinq (35) qui présentent cette situation de non réussite ; vingt-quatre (24) concernant les savoirs explicites et onze (11) les savoirs implicites.

Comme dit précédemment, les échecs sur les savoirs implicites s'expliquent en grande partie par le caractère implicite des savoirs sollicités. C'est ainsi que cinq (5) indicateurs ont été totalement non réussis du fait qu'aucun apprenant ne se souvenait plus du savoir associé comme en témoignent les discussions et les questions pendant le test (exemple : « au fait, c'est quoi le PGCD ? Comment on calcule le PGCD ? »). Quant aux six (6) autres indicateurs de savoirs implicites, moins de la moitié des apprenants ne se

sont pas souvenus de la notion implicite associée à l'indicateur. Dans cette situation, le meilleur indicateur a obtenu huit sur dix-sept (8/17).

Les échecs sur les savoirs explicites nous interpellent particulièrement, c'est donc ce type de savoirs que nous allons, autant que faire ce peu, tenter de comprendre. Les indicateurs qui nous renseignent sur les savoirs explicites peuvent être regroupés en six (6), ce sont :

- Les affectations ;
- Les lectures et écritures (ou affichages) ;
- Les alternatives ;
- Les déclaratives ;
- Les itérations ;
- Les opérateurs.

## ▪ Les affectations

Le cours mentionne principalement deux (2) types d'affectations : les affectations d'expression (ou de calcul) et les affections de valeur.

Dans notre test, nous avons, par soucis de précision éclaté nos indicateurs d'affectation en quatre : trois affectations particulières qui sont l'affectation d'une fonction prédéfinie, l'initialisation et l'incrémentation. Les deux premières sont de valeur alors que l'incrémentation est une affectation de calcul. Dans la dernière affectation, nous logeons toutes celles qui ne sont pas les trois précédentes, qu'elles soient d'expression ou de valeur. C'est cette catégorie qui a été majoritairement non réussie par les apprenants. Nous remarquons que cette affectation pouvait être convoquée dans les quatre exercices proposés lors du test. Elle n'est réussie par aucun apprenant dans le première exercice, réussie par seulement trois(3) dans le quatrième exercice. Cependant, dans le second et le troisième exercice elle est plutôt bien utilisée, respectivement par quatorze (14) et onze (11) apprenants. Dans l'exercice un, l'affectation de calcul de la dernière catégorie (*« affectation expression/valeur »*), pouvait être utilisée pour déterminer le reste de la division euclidienne. La variable ainsi affectée, servant par la suite à poser convenablement la condition associée à l'alternative double indiquant ainsi si la valeur entrée est paire ou impaire.

Nous constatons donc au regard des productions des apprenants qu'ils n'ont pas fait d'affectation pour calculer le reste car pour savoir si un nombre est pair ou impair, il faut calculer le reste entier de la division de ce nombre entier par deux (exemple : 5 ; le reste « **RESTE** » vaut 1 donc 5 « **VALEUR** » est impaire. Ici on note **5 mod 2 = 1**. En utilisant des variables et la syntaxe algorithmique, on aura **RESTE ← VALEUR mod 2**). Ici, l'affectation dont le symbole est (←) permet de déterminer uniquement le reste et par la suite une discussion en fonction de la valeur du reste dira si le nombre est pair (reste

égal à 0) ou impair (reste égale à 1). Mais sur les productions des apprenants, ils ont directement utilisé l'alternative double en lui imposant au moment de poser la condition d'utiliser l'opérateur arithmétique « modulo ».

Exemple :

**Si RESTE ← VALEUR mod 2 = 0, alors**... {*L'utilisation de modulo ici est syntaxiquement incorrecte*}

Cette démarche est correcte d'un point de vu mathématique, mais du fait de la rigueur des règles syntaxiques de l'algorithmique et des langages informatiques, cette écriture n'est pas correcte. En effet, les règles syntaxiques applicables ici sont celles de l'alternative double : « *qui dit que **si** doit être suivi d'une **condition*** … », et celle qui dit que seul « ***les opérateurs logiques et relationnels servent à formuler des conditions*** **»**. C'est donc ici, l'utilisation de l'opérateur arithmétique « modulo » dans l'expression de la formulation de la condition qui pose problème, rendant syntaxiquement incorrecte l'alternative double en l'absence préalable d'une affectation de calcul permettant de déterminer la variable « RESTE » à l'aide de l'opérateur arithmétique « modulo »

On doit donc avoir :

**RESTE←VALEUR mod 2** {*on utilise modulo pour déterminer le reste*}

**Si RESTE = 0 alors…** {*Puis on discute selon la valeur du reste dans l'alternative double*}

**sinon** …

**fsi**.

Pourquoi donc une telle confusion ? Il nous semble évident qu'il y a ici une influence importante des mathématiques sur l'informatique.

Pour l'exercice quatre, il fallait également faire une affectation d'expression. Mais du fait que l'exercice n'a pas été traité par les apprenants à cause des connaissances liées aux savoirs implicites qu'il convoquait, il est difficile de comprendre dans cette production les difficultés spécifiques liées à chaque indicateur. Néanmoins, pour les trois sujets qui ont pu utiliser cette affectation, sans pour autant comprendre l'exercice, il se dégage que cette affectation est plutôt syntaxiquement bien utilisée nonobstant quelques problèmes de sens de l'inégalité dans la condition préalable. Donc pour ces trois sujets, l'exercice n'est pas réussi à cause des savoirs implicites liés à la notion mathématique du PGCD et aux règles de calcul de ce dernier.

- ## Les lectures et écritures (ou affichages)

Dans un algorithme, les instructions qui permettent à l'utilisateur de rentrer des valeurs au clavier pour qu'elles soient utilisées par le programme sont des opérations de **lecture**. Tandis que les instructions qui permettent au programme de communiquer des

valeurs à l'utilisateur en les affichant à l'écran sont des opérations d'**écriture ou** d'**affichage**.

A première vue, on relève comme une contradiction entre ce qui devait s'appeler lecture et qu'on a appelé écriture, et ce qui s'appelle écriture ou affichage et qui devait s'appeler lecture. Mais lorsqu'on y réfléchi, un algorithme, c'est une suite d'instructions qui programme la machine, pas l'utilisateur. Donc quand on dit à la machine de lire une valeur, cela implique que l'utilisateur va devoir écrire cette valeur ou un capteur va l'acquérir à l'extérieur. Et quand on demande à la machine d'écrire une valeur, c'est pour que l'utilisateur puisse la lire. Lecture et écriture sont donc des termes qui comme toujours en programmation, doivent être compris du point de vue de la machine qui sera chargée de les exécuter.

Au cours de l'implémentation d'un algorithme, on réalise deux types de lecture : on peut lire une variable ou plus d'une variable.

Si la lecture d'une variable ne pose pas particulièrement de problème, c'est quand il s'agit de deux ou plus de deux variables qu'il y a à faire un peu attention. La règle syntaxique nous donne deux possibilités : soit on utilise un seul "**LIRE"** suivi entre parenthèse de toutes les variables, séparées les unes des autres par une virgule ; ou on utilise autant "**LIRE"** qu'il y a de variables.

Pendant notre test, l'indicateur de lecture a été utilisé pour lire une variable et pour lire plus d'une variable. Dans le premier cas nous avons eu un taux de réussite de quinze sur soixante-huit (15/68) et la lecture de plus d'une variable dix-huit sur cinquante-un (18/51) soit un pourcentage de réussite pour l'indicateur de lecture de vingt-sept virgule soixante-treize pour cent (27,73%).

Concernant l'écriture, on peut faire l'affichage d'une ou de plusieurs variables ou l'affichage d'un texte (caractère ou chaîne de caractères). Lors de notre test, nous avons utilisé quatre indicateurs d'affichage : l'affichage d'un texte, l'affichage d'une variable, l'affichage de plus d'une variable et enfin l'affichage d'un texte et d'une variable. Nous avons ajouté les deux derniers car bien qu'étant des affichages comme les précédents, ils font tout de même intervenir des règles syntaxiques particulières lors de leur utilisation. Au regard des résultats du test, il en découle que les quatre indicateurs d'affichage sont plutôt mal réussi, ils récoltent des résultats allant de un sur soixante (1/68) pour le moins réussi et vingt-quatre sur soixante-huit (24/68) pour le mieux réussi, soit un pourcentage de réussite de vingt-deux virgule soixante–quatorze pour cent (22,74%). L'affichage de texte est sollicité dans les quatre exercices, il est assez mieux réussi dans les deux premiers exercices que dans les deux seconds avec vingt-quatre sur soixante-huit (24/68). Il en est de même pour l'indicateur d'affichage de variables qui est plutôt bien abordé dans les exercices deux et trois. Les indicateurs d'affichage de plus d'une variable et de texte et variable sont les moins bien réussis. Ils obtiennent respectivement des taux de réussite de neuf sur cinquante-un (9/51) et un sur soixante huit (1/68), soit un pourcentage d'échec cumulé de quatre-vingt-onze virgule soixante pour cent

(91,60%). Au regard des productions des apprenants suite à la tâche qui leur a été confiée, il se dégage trois types d'erreurs : les erreurs d'utilisation de parenthèses (…), d'utilisation du séparateur qui est la virgule (X, Y) et l'utilisation des griffes (**''***erreurs***''**). Toutes ces erreurs étant des erreurs syntaxiques. L'usage des parenthèses est un savoir directement issu du cours. En effet, le cours précise bien que l'affichage d'une variable se fait en utilisant une instruction d'écriture suivi entre parenthèse de la variable à afficher.

Exemple : **écrire (X)**.

Lorsqu'il s'agit d'un caractère ou d'une chaîne de caractère celui-ci est mis entre griffes.

Exemple : **écrire (''aucune solution dans R'')**.

Au-delà de ce non respect d'une règle syntaxique directement énoncé en cours, il y a l'utilisation d'autres règles qui découlent de la compréhension des règles du cours donc des savoirs explicites. Il y a ainsi l'usage de la virgule comme séparateur lors de l'affichage de plus d'une variable ainsi que l'affichage d'une ou plusieurs variables et d'un texte. Il nous semble donc évident que si la règle de base n'est pas comprise celles qui en sont déduites ne sauront nullement être maitrisées. C'est donc logiquement que les indicateurs liés à l'affichage de plus d'une variable et de texte et variable enregistrent les plus grands taux d'échec dans cette catégorie. Nous faisons également le constat qu'un indicateur qui sollicite un savoir explicite est réussi dans un exercice et complètement non réussi dans un autre. C'est le cas de l'affichage d'un texte qui est réussi vingt-une fois par les apprenants sur trente-quatre (21/34) dans les exercices 1 et 2, alors que les apprenants le réussissent seulement trois fois sur trente-quatre (3/34) dans les exercices 3 et 4. On peut ainsi déduire que lorsque l'exercice est complexe, interrogeant des savoirs explicites ou faisant appelle à des savoirs implicites, les apprenants ont du mal à appliquer ou utiliser des savoir-faire qui pourtant ont été précédemment utilisées.

- ## Les alternatives

Au cours de notre test il se dégage que de façon générale, les alternatives ont été très mal réussies. En effet, l'alternative simple n'a été réussie qu'une seule fois sur soixante-huit (1/68). L'alternative double quant à elle n'est réussie que quatre fois sur soixante-huit (4/68). Tandis que l'alternative multiple récolte un score de deux sur soixante-huit (2/68).

Ici, il ressort le constat selon lequel de manière générale, les structures alternatives ne sont pas bien abordées par nos apprenants.

Souvent en algorithmique, les problèmes nécessitent l'étude de plusieurs situations qui ne peuvent pas être traitées par les séquences d'actions simples. En effet, disposant avant l'exécution de plusieurs situations, on ne peut donc pas anticiper sur la situation qui sera exécutée : l'écriture de l'algorithme doit donc prévoir tous les cas possible. Ce

sont les structures conditionnelles ou alternatives qui le permettent, en se basant sur une condition ou un prédicat.

Lors de notre test, nous avons constaté que les apprenants commettent des erreurs de non maitrise de la syntaxe. En effet, la syntaxe impose que le "**SI"** soit suivi de la condition, et l'action précédée de "**ALORS "**, pour les alternatives simple et double. Mais plusieurs apprenants ont oublié de mettre "**ALORS"** après la condition ou encore ils ont mis "**FAIRE"** à la place de "**ALORS"**. De plus, ils ne savent pas clairement exprimer une condition, confondant même des conditions et des expressions.

Exemple :

**Si D ← sqr(B) – 4 x A x C > 0 alors** {*syntaxiquement incorrecte*}

**(Action)**

**Fsi**

Dans l'exemple ci-dessus, **D** ← sqr(**B**) – 4 x **A** x **C** est une expression qu'il convient d'exécuter avant d'utiliser son résultat pour poser le prédicat. Pour poser un prédicat, on utilise un opérateur logique ou relationnel avec une variable.

Exemple :

**Si D > 0 alors** {*syntaxiquement correct*}

**(Action)**

**Fsi**

Concernant l'alternative multiple qui ont été échouées soixante et six fois sur soixante-huit, elle est introduite par "**SELON QUE"** suivi d'une première condition, viennent ensuite un ou plusieurs "**OU QUE"** suivi des autres conditions. Pour terminer, un "**AUTREMENT"** introduit l'action qui est effectuée lorsque toutes les conditions indiquées ne sont pas remplies. Là encore, au lieu de mettre "**FAIRE"** après les conditions, les apprenants utilisent "**ALORS"** comme pour les alternatives simples ou doubles.

Un autre type d'erreurs est la confusion entre l'usage d'une structure conditionnelle et une structure répétitive ou itérative. Pour beaucoup d'apprenants, les boucles effectuent exactement les mêmes opérations que les alternatives. Ils confondent donc par exemple :

**Si D > 0 alors …**

Avec

**Tant que D > 0 faire …**

On pourrait croire à priori que si la nature humaine était logique, une condition ne peut pas être une répétition. Ce qui voudrais dire que le résultat lors du traitement d'une instruction conditionnelle ne peut pas être le même que celui d'une itération. Comme nous le verrons plus bas, les structures répétitives ou itératives permettent d'exécuter plusieurs fois de suite une ou plusieurs instructions, elles offrent une économie d'écriture, contrairement aux structures alternatives qui offrent la possibilité d'effectuer un choix selon une ou plusieurs conditions. Alors comment expliquer une telle confusion ? Il nous semble que cela est causé par un mauvais apprentissage, c'est-à-dire que le sujet raisonnant très peu, lors de l'accomplissement de la tâche, il active des mauvaises connaissances, notamment celles liées à l'itération alors qu'il est face une situation d'alternative.

- **Les déclaratives**

Dans notre test, concernant l'indicateur relatif à la déclaration des variables, nous avons utilisé uniquement la déclaration du type numérique (entier et réel) avec un taux de réussite de trente-trois sur soixante-huit (33/68). Dans l'exercice 1 et l'exercice 3, la déclaration des variables est quasiment non réussie par l'ensemble des sujets avec respectivement zéro sur soixante-huit (0/68) et trois sur soixante-huit (3/68).

Au cours de notre expérimentation, la principale erreur commise par les apprenants était le fait de ne pas déclarer des variables. En effet, dans beaucoup de rendus nous avons constaté l'utilisation de plusieurs variables sans en faire la déclaration. Cette précaution syntaxique importante permet de préparer des emplacements en mémoire car dans un ordinateur, une information est toujours stockée dans une mémoire.

Outre la réservation de l'emplacement en mémoire, la déclaration de variables doit aussi tenir compte de la taille de la mémoire, cette précaution reste importante en dépit de l'évolution technologique qui permet actuellement de disposer d'espace-mémoire plus important car un choix adéquat de la taille de la mémoire rendra toujours un programme plus rapide qu'un mauvais choix. Le programmeur doit anticiper sur la nature des éléments à stocker, sur les valeurs que prendra la variable déclarée. C'est donc le « type » de la variable, c'est le « type » qui détermine la taille de l'espace mémoire à réserver, mais aussi conditionner la manière dont la machine va interpréter le contenu de l'emplacement-mémoire pour effectuer les traitements. Certains apprenants ont aussi, lors de la déclaration des variables, omis de préciser le type de la variable ou déclarer le mauvais type. Une confusion de type aura pour conséquence une réservation d'un emplacement mémoire soit trop petit, soit trop grand et surtout inapproprié pour la valeur de la variable contenue dans les instructions du programme. Cette situation peut être plus grave lorsque l'on teste sur le reste d'une division. D'une manière générale, toutes ces contraintes liées au formalisme dans la partie déclarative n'empêchent pas le corps de l'algorithme d'être correct, c'est-à-dire qu'il soit capable de résoudre le problème posé. Dans le cadre d'un enseignement-apprentissage d'un formalisme qui tient compte de la partie déclarative, on se projette dans la traduction d'un algorithme en un langage de programmation, lequel langage impose dans sa

syntaxe une partie déclarative. Dans le cadre de l'utilisation d'un logiciel de simulation d'algorithme comme AlgoBox, au lancement nous trouvons sur la fenêtre de travail trois titres (VARIABLES, DEBUT_ALGORITHME et FIN_ALGORITHME) et dans l'onglet « opérations standards », le bouton « nouvelle variable » actif. On lit dans la partie aide en ligne de l'application le texte : « *Pour utiliser une variable, il faut d'abord la déclarer (bouton "Déclarer nouvelle variable"). Pour ajouter un nouvel élément à l'algorithme, il faut d'abord insérer une nouvelle ligne (bouton "Nouvelle Ligne"), puis cliquer sur un des boutons disponibles dans le panneau "Ajouter code" »*. A ce niveau, nous comprenons que la première étape de la simulation d'un algorithme sur cette application, c'est de déclarer les variables de l'algorithme. Dans cette première étape, nous nommons nos variables que nous avons au préalable choisies puis le type est sélectionné sur un menu constitué de trois articles (NOMBRE, CHAINE et LISTE) qui nous est proposé par l'application. C'est donc après la déclaration d'au moins une variable qu'il est possible d'ajouter une ligne de programme par l'intermédiaire du bouton « nouvelle ligne ».

## ▪ Les itérations (ou boucles)

Les structures itératives représentent la partie la plus délicate pour l'apprenti programmeur, car autant il est assez facile de comprendre comment fonctionnent les boucles, autant il est souvent long d'acquérir les réflexes qui permettent de les élaborer judicieusement pour traiter un problème donné. On peut dire en fait que les boucles constituent, avec les structures conditionnelles les seules vraies structures logiques caractéristiques de la programmation : cela n'existe que dans les langages de programmation proprement dits.

D'une manière simple, on peut dire que les structures de contrôle itératives (boucles) permettent d'exécuter plusieurs fois de suite une ou plusieurs instructions. Il en existe trois (3) distinctes :

La boucle "**POUR"** : La structure "**POUR"** permet d'exécuter une instruction un nombre connu de fois.

La boucle "**REPETER"** : La structure "**REPETER"** est employée dans les situations où l'on doit procéder à un traitement systématique sur les éléments d'un ensemble dont on ne connait pas d'avance le nombre d'exécutions.

La boucle "**TANTQUE"** : Comme la précédente, la structure "**TANTQUE"** est employée dans les situations où l'on doit procéder à un traitement systématique sur les éléments d'un ensemble dont on ne connait pas d'avance le nombre d'exécutions.

Le choix de la structure de contrôle se fait en fonction du nombre d'itérations à effectuer. Quand le nombre d'itérations est déterminé à l'avance, une boucle "**POUR"** est la plus appropriée. Sinon, la boucle "**RÉPÉTER"** est la plus appropriée lorsque la condition de boucle ne peut être évaluée avant la première itération, ou si au moins une itération doit être exécutée. Dans le cas contraire, il est plus judicieux d'utiliser une boucle "**TANT QUE"**.

Lors de notre test, ces structures ont été sollicitées, cependant les scores réalisés nous interpellent à plus d'un titre. En effet, les structures itératives avec test à l'entrée "**TANT QUE"** ont le meilleur taux de réussite à savoir huit sur trente-quatre (8/34) tandis que toutes les autres ont été totalement échouées par l'ensemble des sujets et ce quelque soit l'exercice. Ce qui, au total, représente un pourcentage de réussite pour l'ensemble des structures itératives et imbrications contenant des itérations de cinq virgule vingt-deux pour cent (5,22 %).

Pour les apprenants qui ont utilisé ces structures itératives, la principale erreur commise était la non maitrise de la syntaxe associée à chacune des structures que le sujet souhaite utiliser pour son algorithme. Au risque d'être redondant, toutes les erreurs liées à la confusion entre les structures alternatives et les structures itératives, ainsi que la confusion entre une condition et une expression se retrouvent ici. Nous n'y reviendrons donc pas.

## ▪ Les opérateurs

Un opérateur est un signe qui relie deux valeurs, pour produire un résultat.

Les opérateurs possibles dépendent du type des valeurs qui sont en jeu. Accompagnés d'identificateurs ou de valeurs, ils permettent de construire des expressions.

Il y a des opérateurs numériques (ou arithmétiques), alphabétiques, logiques (ou booléens) et relationnels.

Pendant la première expérimentation, nous avons constaté que les opérateurs relationnels ont été souvent utilisés. Ils ont obtenu un score de trente-trois sur soixante-huit (33/68). Quant aux opérateurs arithmétiques classiques, c'est-à-dire l'addition, la soustraction et la multiplication, ils ont été réussis à vingt-sept sur cinquante et un (27/51). Il est à signaler qu'un seul apprenant a essayé d'utiliser sans succès un opérateur logique. La division sous ses deux formes représente l'opérateur arithmétique qui visiblement a constitué le talon d'Achille des apprenants. En effet, l'opérateur modulo qui donne le reste d'une division de deux entiers n'a pas été convenablement utilisé et réussi par les participants au test. Concernant la division classique, l'erreur syntaxique courante commise par les apprenants était l'utilisation de la barre horizontale **(___)** au lieu de la barre oblique **(/)** pour séparer le dividende du diviseur.

En conclusion, sur l'ensemble des indicateurs et l'ensemble des exercices et des apprenants, nous arrivons à un pourcentage d'échecs de soixante-onze virgule quatre pour cent (71,4%) comme l'illustre le graphique de la figure 20 ci-dessous, confirmant ainsi notre constat du fort taux d'échecs dans ce module consacré à l'algorithmique à l'ENSET de Libreville.

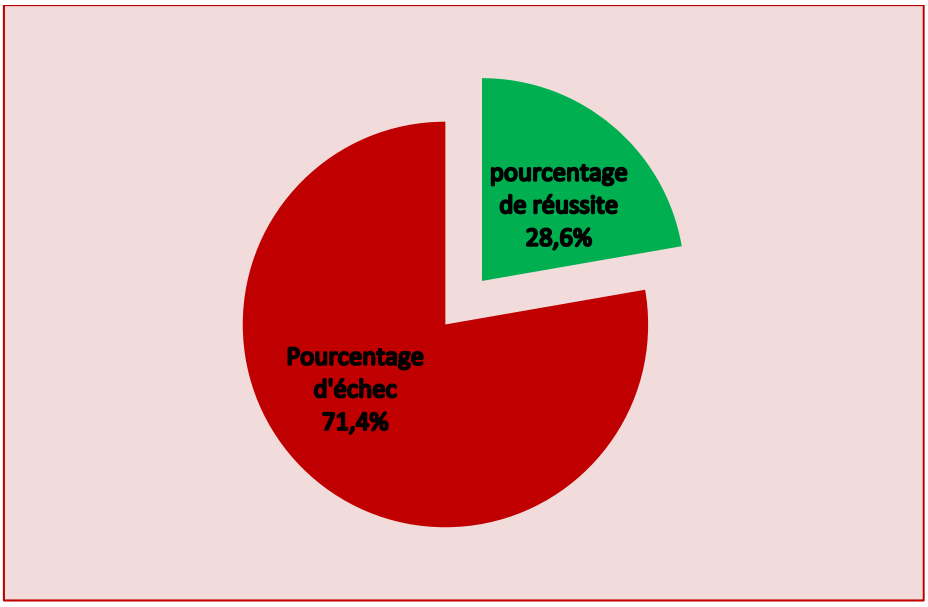



*Figure 20: répartition des pourcentages d'échec et de réussite*

De ce taux d'échec, plusieurs enseignements peuvent être tirés.

Nous pouvons dans un premier temps identifier les indicateurs sur lesquels les apprenants ont le plus de difficulté, ceux qui recueillent le plus faible pourcentage de réussite. Ils représentent trente-cinq indicateurs sur quarante-quatre (35/44). Les indicateurs relatifs aux savoir-faire explicites donc déduits directement des notions de cours sont les moins biens réussis. Et enfin qu'un savoir implicite mal appris, c'est-à-dire que l'apprenant active la mauvaise connaissance pour résoudre un problème conduit indubitablement vers l'échec à l'exercice qui sollicite ce savoir et ce, même si les savoir-faire à mettre en œuvre sont connus de l'apprenant.

# 3.2. Présentation de la deuxième expérimentation

## 3.2.1. Dispositif et schéma expérimental

Dans le cadre de notre deuxième expérimentation, nous avons utilisé un plan de recherche avec groupe témoin non équivalent. En effet, ce type de plan est particulièrement pratiqué lorsqu'il n'est pas possible d'assigner aléatoirement les participants aux différentes conditions de la recherche et qu'il est possible de se replier sur un groupe de comparaison qui n'est cependant pas nécessairement équivalent au groupe soumis au traitement, à l'évènement ou à la situation (Simeone, 2006).

Ainsi, le plan de type pré-test et post-test avec groupe témoin non équivalent a été le suivant (voir tableau 5) :

| **Pré-test** | | **Post-test** |
|---|---|---|
| Groupe expérimental (GE) | Intervention (TP AlgoBox) | Groupe expérimental (GE) |
| Groupe témoin (GT) | | Groupe témoin (GT) |

*Tableau 5 : plan expérimental de type pré-test et post-test avec groupe témoin*

Le plan comprend deux (2) groupes distincts, l'un ayant été exposé à l'utilisation du nouvel outil (GE) et l'autre pas (GT). Ces deux (2) groupes étant évalués avant et après l'intervention. Ces groupes sont considérés comme non équivalents parce que non constitués sur la base d'une répartition aléatoire.

Ce plan permet de mesurer l'importance du changement entre un pré-test et un post-test, selon que le groupe de participants ait été ou non exposé à l'intervention. L'analyse des résultats nécessite donc une triple comparaison :

- entre le GE et le GT ;
- entre le pré-test et le post-test ;
- entre les écarts constatés entre les groupes au pré-test et ceux observés entre les groupes au post-test.

Il est important de signaler que la présence du groupe témoin et d'une évaluation au pré-test permet de mieux répondre au problème de validité interne soulevée par les protocoles pré-expérimentaux. Toutefois, le fait que les groupes n'aient pas été constitués sur la base d'une assignation aléatoire à partir de la même population, c'est-à-dire qu'ils soient non équivalents peut poser des problèmes d'interprétation. A cet égard, la comparaison entre le groupe expérimental et le groupe témoin au pré-test est particulièrement importante car elle permet de démontrer que l'échantillon est homogène au départ. Plus nous pourrons montrer que les groupes expérimental et témoin sont semblables au pré-test, plus le protocole de notre recherche sera considéré comme valide sur le plan interne.

La technique de l'échantillonnage par quotas ou les procédures d'appariement visant à constituer des groupes similaires sur certaines dimensions peuvent permettre à priori d'accentuer la proximité des groupes.

Nous avons donc adapté ce plan expérimental à notre recherche conformément au schéma de la figure 21 ci-dessous.

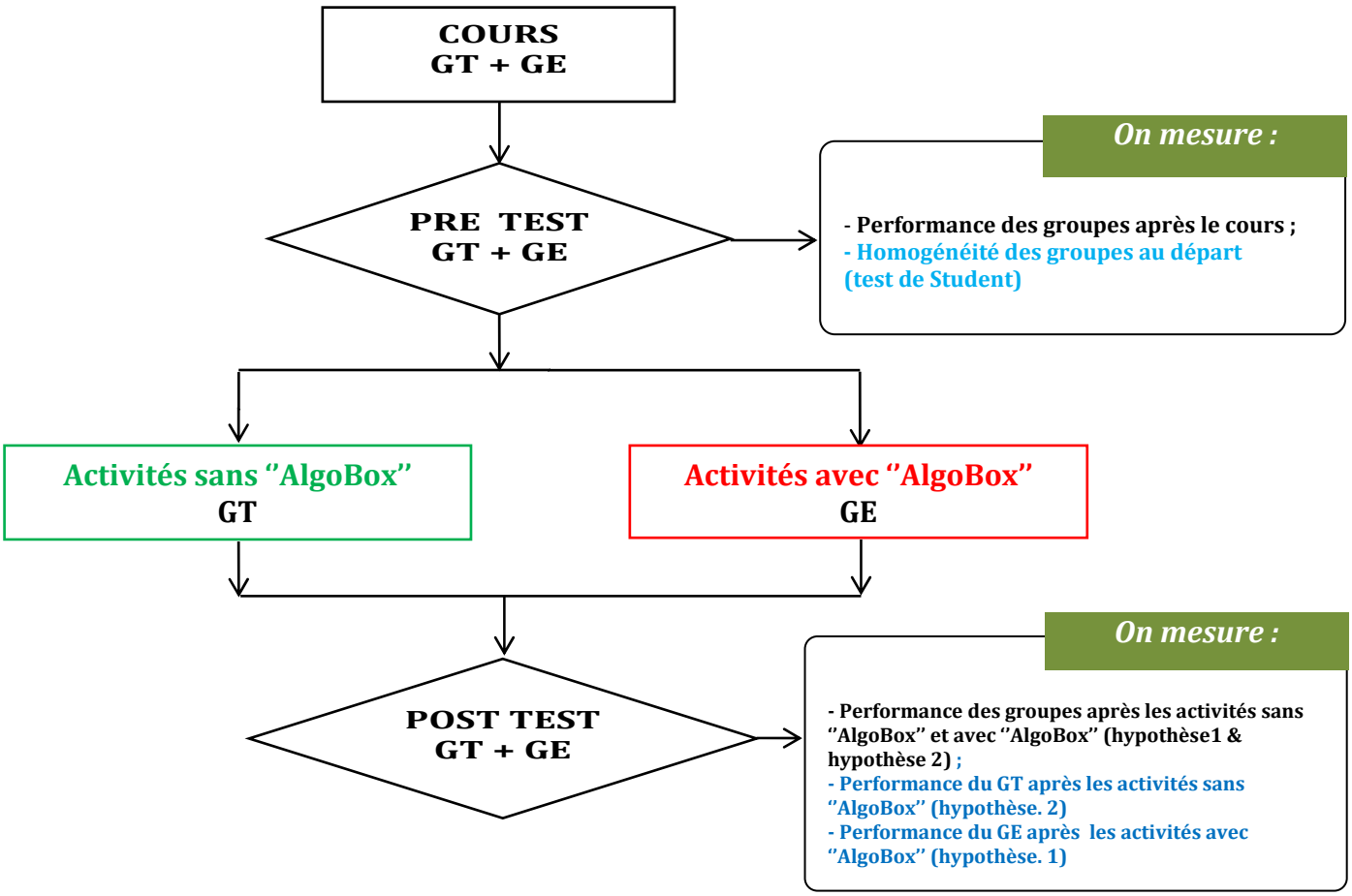



*Figure 21 : schéma du plan expérimental.*

Par contre, par souci d'objectivité, nous ne manquerons pas de souligner les inconvénients d'un tel dispositif. En effet, cette procédure ne tient compte que de potentielles différences observées au pré-test et non de la possibilité que ces différences s'accentuent avec le temps, en dehors de tout effet de l'intervention. Trois (3) types de biais sont généralement observés dans la relation entre la variable indépendante et la variable dépendante.

Tout d'abord, il est possible que les participants d'un groupe deviennent plus compétents dans la dimension évaluée par la variable dépendante que les participants des autres groupes pendant la période de l'intervention et ce indépendamment de cette intervention. Ce phénomène peut intervenir soit à cause d'une interaction maturation, soit à cause d'une interaction sélection-expérience (Simeone, 2006). Dans le premier cas les deux groupes de participants peuvent montrer des courbes d'évolution ou de développement divergentes dues à des facteurs endogènes les distinguant, ce qui

pourrait rendre compte partiellement ou entièrement d'un changement plus important dans un groupe que dans un autre. Dans le second cas, les participants d'un groupe peuvent avoir été exposés au cours de l'interaction à des facteurs exogènes, tel que des expériences rationnelles stimulantes ou valorisantes, autres que celles prévues par le traitement et qui au final pourraient expliquer la progression plus importante de ce groupe par rapport à l'autre (Simeone, 2006).

Enfin il est clair que ces phénomènes interactifs sont plus susceptibles d'apparaitre lorsque l'intervalle entre le pré-test et le post-test est grand.

Ensuite, il est possible que le caractère non équivalent des groupes étudiés produise un biais de régression statistique différentielle c'est-à-dire que la régression statistique à la moyenne peut se manifester de façon plus marquée dans un groupe que dans l'autre.

Enfin, il est possible que l'outil choisi pour estimer l'amplitude du changement, la variable dépendante, puisse être plus ou moins apte à différentier les individus selon leur niveau ou leur position dans la dimension évaluée (Shuttelworth, 2009).

## 3.2.2. Échantillonnage

Dans le cas de notre recherche, notre échantillon était constitué de tous les étudiants de la première année de l'ENSET du cycle des sciences et techniques industrielles. Ce sont des bacheliers technologiques provenant de tous les lycées techniques du pays et recrutés sur la base d'un concours. Donc notre cadre d'étude a lui-même fixé son échantillon. Ce qui nous impose, pour ainsi dire, notre échantillon. Cependant, pour respecter le schéma expérimental que nous avons adopté, nous avons néanmoins réparti les sujets en deux groupes : un groupe expérimental (GE) et un groupe témoin (GT). Cette répartition quant à elle s'est faite en appliquant la technique d'échantillonnage dite « des quotas » car il nous fallait avoir un contrôle sur la représentativité.

D'une manière générale, l'application de la méthode des quotas nécessite une connaissance préalable des caractéristiques de la population. Cette méthode va consister, à partir des informations concernant la population d'origine, à extraire un échantillon en respectant les proportions des différentes caractéristiques de la population d'origine que l'on soupçonne pouvoir avoir un effet sur le phénomène étudié (Shuttelworth, 2009). Dans notre cas, c'est pour effectuer la répartition des sujets dans les deux groupes que nous avions utilisé cette technique. En effet, il est important par rapport à notre étude que les sujets soient affectés dans chacun des groupes en respectant certaines proportions en fonction des critères que nous avions identifiés et ayant une relation objective avec notre recherche. Cette relation dite objective ne suffisant pas, la pertinence de cette constitution de groupe doit donc être vérifiée scientifiquement en démontrant l'homogénéité de départ des deux groupes.

Notre échantillon est constitué de trente neuf (39) sujets répartis en deux (2) groupes :

- Un groupe expérimental (GE) : vingt (20) sujets ;

- Un groupe témoin (GT) : dix neuf (19) sujets.

La répartition des sujets dans les différents groupes s'est faite selon quatre (4) critères qui sont par ordre d'importance :

- Critère 1 : le type de baccalauréat technologique (BT- F1- F1D- F3- MI- MVA- F2- F4).

- Critère 2 : le passé algorithmique (avoir déjà fait ou pas de l'algorithmique).

- Critère 3 : le sexe (M ou F).

- Critère 4 : l'âge du sujet (la moyenne d'âge : 23 ans pour le groupe témoin et 22,5 ans pour le groupe expérimental).

Le tableau 6 ci-dessous décrit la répartition des sujets par groupe en fonction des différents critères.

| | **critère 1** | | | | | | | | **critère 2** | | **critère 3** | | **critère 4** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **BT** | **MI** | **MVA** | **F1D** | **F1** | **F2** | **F3** | **F4** | **oui** | **non** | **M** | **F** | **Moyenne** |
| **Groupe témoin (GT)** | 0 | 3 | 3 | 3 | 4 | 1 | 5 | 0 | 4 | 15 | 15 | 4 | 23 |
| **groupe expérimental (GE)** | 1 | 6 | 1 | 2 | 3 | 1 | 5 | 1 | 3 | 17 | 15 | 5 | 22,5 |
| **Échantillon** | **1** | **9** | **4** | **5** | **7** | **2** | **10** | **1** | **7** | **32** | **30** | **9** | **22,75** |

*Tableau 6 : Répartition des sujets par groupe*

D'une manière générale, pour notre recherche, dans un premier temps, nos démonstrations vont s'appuyer sur la variable de statistique descriptive qui est la moyenne. Cependant, pour prouver l'homogénéité de nos groupes, on va adjoindre à la moyenne d'autres variables comme l'écart type et/ou la variance.

## 3.2.3. Instruments d'investigation

Notre seconde expérimentation s'est déroulée l'année académique suivant la première expérimentation. Nous étions donc non seulement au cours d'une nouvelle année scolaire, mais nous avions aussi un nouveau public. De ce faite, au cours du pré-test, nous avons utilisé le même instrument d'investigation qu'à la première expérimentation.

Quant au post-test, dans le but de vérifier notre second hypothèse de recherche, chaque exercice était suivi de quelques indications dont l'unique but était d'énoncer (réactivation) des connaissances liées aux savoirs implicites portées par la tâche prescrite, et qui seront sollicitées dans l'élaboration de la solution, afin de faciliter chez l'apprenant la compréhension de la situation-problème comme le stipule ladite hypothèse.

## 3.2.4. Résultats de la deuxième expérimentation

### 3.2.4.1. Homogénéité de l'échantillon

Il est question ici, après avoir fait la répartition des sujets dans les deux groupes selon la technique des quotas, en s'appuyant sur nos quatre critères, de démontrer que les deux groupes ainsi obtenus sont homogènes. En effet, dans le cadre de notre expérimentation, nous avons effectué des mesures répétées, c'est-à-dire avant et après l'utilisation de notre artefact computationnel sur les deux groupes qui constituent notre échantillon : le groupe expérimental (GE) est celui qui a utilisé l'artefact entre les deux mesures au pré test et au post test, le groupe témoin (GT) ne s'est pas servi du logiciel. Nous allons utiliser la mesure au pré test des deux groupes pour nous renseigner sur leur homogénéité. Nous pouvons donc comparer les mesures des moyennes de ces deux groupes pour démontrer qu'ils sont homogènes ou pas. En effet, les groupes seront homogènes si leurs performances, ici les moyennes sont sensiblement identiques entre elles.

Afin de renforcer ce résultat, on peut aussi utiliser un test de comparaison de deux moyennes. Du fait de la faiblesse de notre échantillon, de la variabilité de deux moyennes provenant de deux groupes indépendants, le test t de Student semble être le plus adapté. En effet, nous allons tester l'hypothèse nulle à partir de deux moyennes provenant de deux groupes indépendants. Nous allons en fait estimer si deux moyennes populationnelles (performance obtenue aux exercices) sont égales en nous basant sur le résultat de la comparaison entre ces deux groupes, ces populations sont identiques (homogènes).

Pour utiliser ce test, il est important de formuler une hypothèse zéro, dite hypothèse nulle, qui stipule dans notre cas qu'il n'y a pas de différence entre les moyennes des deux groupes. En d'autres termes, la différence entre les deux moyennes des deux groupes est de zéro (0). Ainsi, le test devrait conclure au non rejet de cette hypothèse nulle pour nous amener à déduire que nos groupes sont homogènes.

**Tableau de valeurs**

| | **Pré test** | |
|---|---|---|
| | **Moyenne** | **Écart type** |
| **GT** | 18,16 | 11,77 |
| **GE** | 21,3 | 10,77 |

*Tableau 7 : valeurs de la moyenne et de l'écart-type au pré-test*

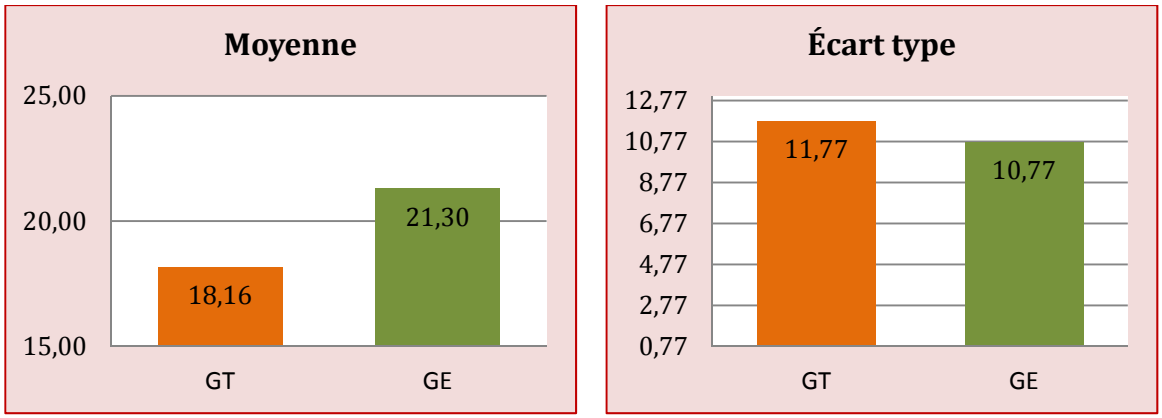



*Figure 22 : graphiques de la moyenne et de l'écart type des deux groupes au pré-test*

**Interprétation**

Nous avons expliqué précédemment que le GT et GE sont homogènes si :

Moyenne GE est égale à la moyenne GT.

Ces conditions étant difficilement réalisables à cause de notre technique d'échantillonnage, mais aussi à cause de la variable même que nous utilisons : c'est-à-dire la moyenne qui est calculée sur la base des performances de sujets humains. Ces performances peuvent donc être difficilement identiques sans un intervalle de tolérance.

Nos résultats nous donnent une différence de trois virgule quatorze (3,14) entre la moyenne du groupe expérimental (21,30) et celle du groupe témoin (18,16) comme le montre l'histogramme en figure 22 ci-dessous.

Ainsi, nous avons entre les moyennes un écart très faible de l'ordre de 3 points sur 1734.

1734 étant le nombre de fois que les 44 indicateurs peuvent être utilisés par les 39 sujets dans les 4 exercices.

A priori, nous pouvons conclure que le groupe expérimental et le groupe témoin sont homogènes au départ de notre expérimentation.

Pour mieux assoir cette conclusion, il nous semble important d'effectuer des tests statistiques appropriés. En effectuant un test de comparaison de moyenne (test de Student) et de variance (test de Ficher), nous avons les résultats suivants :

**Test t de Student**

- **Statistiques descriptives :**

| Variable | Observations | Obs. avec données manquantes | Obs. sans données manquantes | Minimum | Maximum | Moyenne | Écart-type |
|---|---|---|---|---|---|---|---|
| GE | 20 | 0 | 20 | 0,000 | 51,000 | 21,300 | 10,766 |
| GT | 19 | 0 | 19 | 0,000 | 39,000 | 18,158 | 11,772 |

*Tableau 8 : tableau de statistiques descriptives du pré-test*

- **Test t pour deux échantillons indépendants / Test bilatéral :**

Intervalle de confiance à 95% autour de la différence des moyennes : [-4,171;10,455[

| | |
|---|---|
| Différence | 3,142 |
| t (Valeur observée) | 0,871 |
| \|t\| (Valeur critique) | 2,026 |
| DDL | 37 |
| p-value (bilatérale) | 0,390 |
| Alpha | 0,05 |

*Tableau 9 : tableau de statistiques du test bilatéral au pré-test*

- **Interprétation du test :**

$H_0$ : La différence entre les moyennes est égale à 0.
Ha : La différence entre les moyennes est différente de 0.
Étant donné que la p-value calculée est supérieure au niveau de signification seuil alpha=0,05 comme on le voit au tableau 9 ci-dessus et à la figure 23, *on ne peut pas rejeter l'hypothèse nulle $H_0$.*

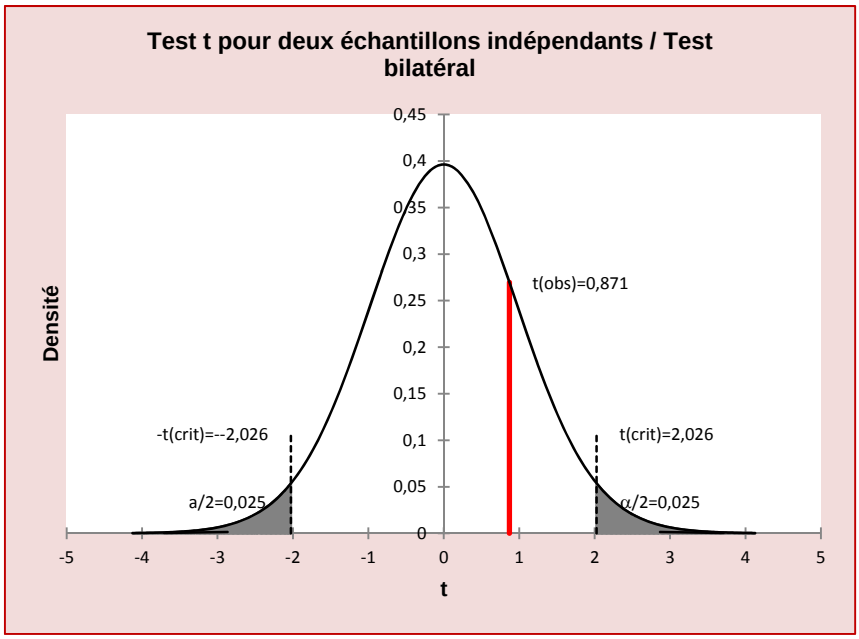



*Figure 23 : graphique du test bilatéral au pré-test (Test t)*

## Test F de Fisher

- **Statistiques descriptives :**

| Variable | Observations | Obs. avec données manquantes | Obs. sans données manquantes | Minimum | Maximum | Moyenne | Écart-type |
|---|---|---|---|---|---|---|---|
| GE | 20 | 0 | 20 | 0,000 | 51,000 | 21,300 | 10,766 |
| GT | 19 | 0 | 19 | 0,000 | 39,000 | 18,158 | 11,772 |

*Tableau 10 : tableau de statistiques descriptives au pré-test (test F)*

- **Test F de Fisher / Test bilatéral :**

Intervalle de confiance à 95% autour du rapport des variances :] 0,325; 2,129 [

| | |
|---|---|
| Rapport | 0,836 |
| F (Valeur observée) | 0,836 |
| \|F\| (Valeur critique) | 2,576 |
| DDL1 | 19 |
| DDL2 | 18 |
| p-value (bilatérale) | 0,701 |
| Alpha | 0,05 |

*Tableau 11 : tableau de statistiques du test bilatéral au pré-test (test F)*

- **Interprétation du test :**

$H_0$ : Le rapport entre les variances est égal à 1.
$H_a$ : Le rapport entre les variances est différent de 1.
Étant donné que la p-value calculée est supérieure au niveau de signification seuil alpha=0,05 comme le montre le tableau 11 ci-dessus et la figure 24, *on ne peut pas rejeter l'hypothèse nulle $H_0$*.

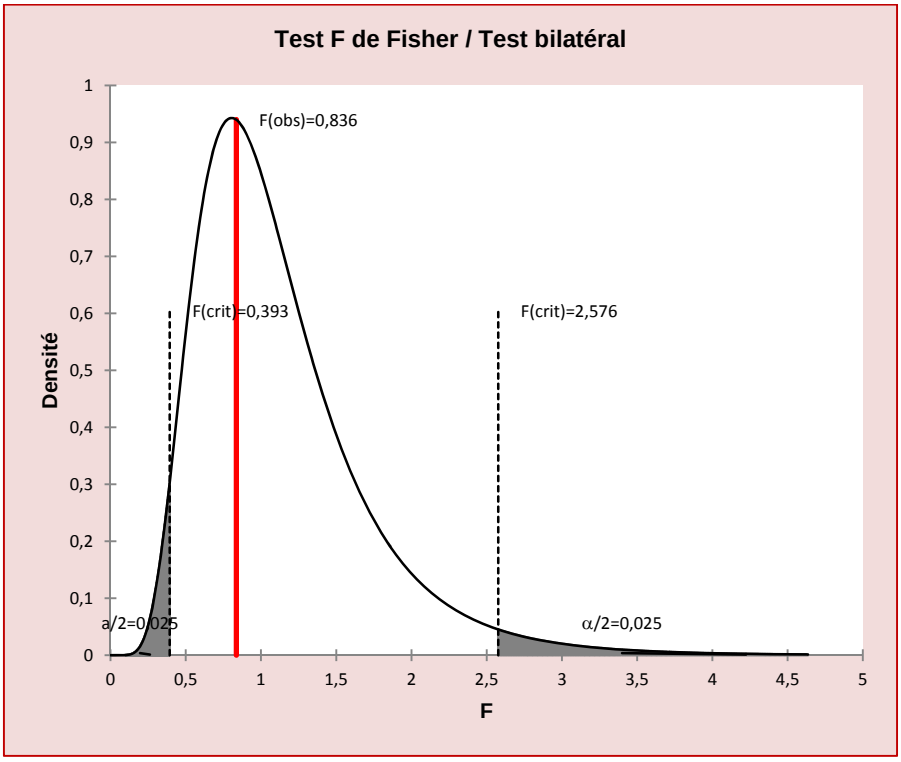



*Figure 24 : graphique du test bilatéral au pré-test (Test F)*

En conclusion, dans ces deux tests, on voit que la p-value calculée est supérieure au niveau de signification seuil alpha de 0,05. Cet information signifie qu'on ne peut pas rejeter l'hypothèse nulle $H_0$. En effet, Si t appartient à $[-t_\alpha;+t_\alpha]$, on accepte $H_0$, qui stipule que nos groupes sont identiques.

- Le test t donne pour $H_0$ : la différence entre les moyennes est égale à 0. Ce qui peut se traduire par l'équation :

$$\overline{Moy}\ \mathbf{GE} - \overline{Moy}\ \mathbf{GT} = 0 \Rightarrow \overline{Moy}\ \mathbf{GE} = \overline{Moy}\ \mathbf{GT}.$$

Le test t de Student nous montre que les moyennes de nos deux groupes sont égales.

- Le test F donne pour $H_0$ : le rapport entre les variances est égal à 1. Ce résultat peut se traduire sous la forme d'une équation par :

$$\mathbf{1} = \frac{\mathbf{VAR\ (GE)}}{\mathbf{VAR\ (GT)}} \Rightarrow VAR\ (GE) = VAR(GT)$$

Si le rapport des variances est égal à 1 alors les deux variances sont égales.

Finalement, nous avons pour ces deux tests, des moyennes égales pour le test t et des variances égales pour le test F pour nos deux groupes. On en déduit donc que ces groupes sont homogènes. Ce qui signifie que les deux groupes ont été constitués de telle sorte que les sujets qui les constituent sont sensiblement de même niveau en algorithmique avant l'utilisation du logiciel AlgoBox et avant l'activation des connaissances liées aux savoirs implicites portés par la tâche prescrite.

### 3.2.4.2. Vérification de la première hypothèse

Cette première hypothèse stipule qu' : « en s'appuyant sur les travaux relatifs à la charge cognitive, l'utilisation d'un outil comme AlgoBox devrait réduire la charge cognitive et de ce fait ***faciliter l'apprentissage de l'algorithmique*** en soulageant en partie l'apprenant de l'activité liée à l'apprentissage du formalisme par réduction de la charge cognitive intrinsèque (Paas, 1994; Sweller, 2010) ».

Lorsque nous analysons notre dispositif expérimental, nous constatons que les résultats au post test des deux groupes sont importants pour la vérification de cette première hypothèse. Donc si nous comparons au post test les performances du groupe ayant utilisé AlgoBox dans ses activités (GE) et celles du groupe ne l'ayant pas utilisé on peut conclure sur un effet d'AlgoBox sur l'apprentissage. Plus précisément, si le groupe qui a utilisé l'artefact computationnel est plus performant que l'autre, on peut dire que l'artefact facilite l'apprentissage. Pour ce faire, nous pouvons comparer les moyennes des deux groupes au post test.

Pour mieux assoir cette démarche à cause de la faiblesse de notre échantillon et de la variabilité des deux moyennes provenant des deux groupes indépendants, nous allons procéder à un test t de comparaison de moyennes.

Contrairement à sa précédente utilisation qui consistait à tester l'hypothèse nulle ($H_0$) à partir de deux moyennes provenant de deux groupes indépendants. Le but était dans ce cas de monter l'égalité des moyennes des deux groupes afin de conclure sur leur homogénéité. Cette fois, c'est hypothèse alternative ($H_a$) qui nous va nous intéresser. Cette hypothèse dit qu'il y a une différence entre les deux moyennes des deux groupes. En effet, le test de Student, ou test *t*, est un ensemble de tests d'hypothèse paramétriques où la statistique calculée suit une loi de Student lorsque l'hypothèse nulle ($H_0$) est vraie. Un test de Student peut être utilisé notamment pour tester statistiquement l'hypothèse d'égalité de l'espérance de deux variables aléatoires suivant une loi normale et de variance inconnue. Il est aussi très souvent utilisé pour tester la nullité d'un coefficient dans le cadre d'une régression linéaire.

Dans notre cas, le test devrait nous permettre de rejeter l'hypothèse nulle pour retenir notre hypothèse alternative. En effet, si nos groupes sont différents, cela signifie que leurs moyennes respectives sont différentes, donc leurs performances sont différentes. Cependant, la différence entre les performances ne nous renseigne pas sur l'identité du groupe qui a réalisé la meilleure performance. Cette information est très importante pour la validation de notre hypothèse de travail car nous avons postulé implicitement que le groupe utilisant l'artefact aura de meilleures performances. Donc nous devons, non seulement démontrer que les moyennes des groupes sont différentes, mais aussi que celle du groupe qui a fait des activités avec le logiciel est supérieure à celle du groupe qui ne s'est pas servi de l'application. Pour montrer la seconde partie, il suffit de calculer et de comparer les moyennes des deux groupes.

Ainsi, le test devrait conclure au rejet de cette hypothèse nulle donc à retenir l'hypothèse alternative et l'observation de la statistique descriptive nous emmènera à voire le groupe qui a la plus grande moyenne. Ces deux éléments étant suffisants pour vérifier notre hypothèse de travail.

**Test t de Student**

- **Statistiques descriptives :**

| Variable | Observations | Obs. avec données manquantes | Obs. sans données manquantes | Minimum | Maximum | Moyenne | Écart-type |
|---|---|---|---|---|---|---|---|
| GE | 20 | 0 | 20 | 29,000 | 63,000 | 47,250 | 10,031 |
| GT | 19 | 0 | 19 | 6,000 | 59,000 | 33,842 | 16,483 |

*Tableau 12 : tableau de statistiques descriptives au post-test (Test t)*

- **Test t pour deux échantillons indépendants / Test bilatéral :**

Intervalle de confiance à 95% autour de la différence des moyennes : [4,607; 22,209[

| | |
|---|---|
| Différence | 13,408 |
| t (Valeur observée) | 3,087 |
| \|t\| (Valeur critique) | 2,026 |
| DDL | 37 |
| p-value (bilatérale) | 0,004 |
| alpha | 0,05 |

*Tableau 13 : tableau de statistiques du test bilatéral au post-test (Test t)*

- **Interprétation du test :**

$H_0$ : La différence entre les moyennes est égale à 0.
$H_a$ : La différence entre les moyennes est différente de 0.
Étant donné que la p-value calculée est inférieure au niveau de signification alpha=0,05 comme observé au tableau 13 ci-dessus , et à la figure 25 *on doit rejeter l'hypothèse nulle $H_0$*, et *retenir l'hypothèse alternative $H_a$.*
Le risque de rejeter l'hypothèse nulle $H_0$ alors qu'elle est vraie est inférieur à 0,38%.

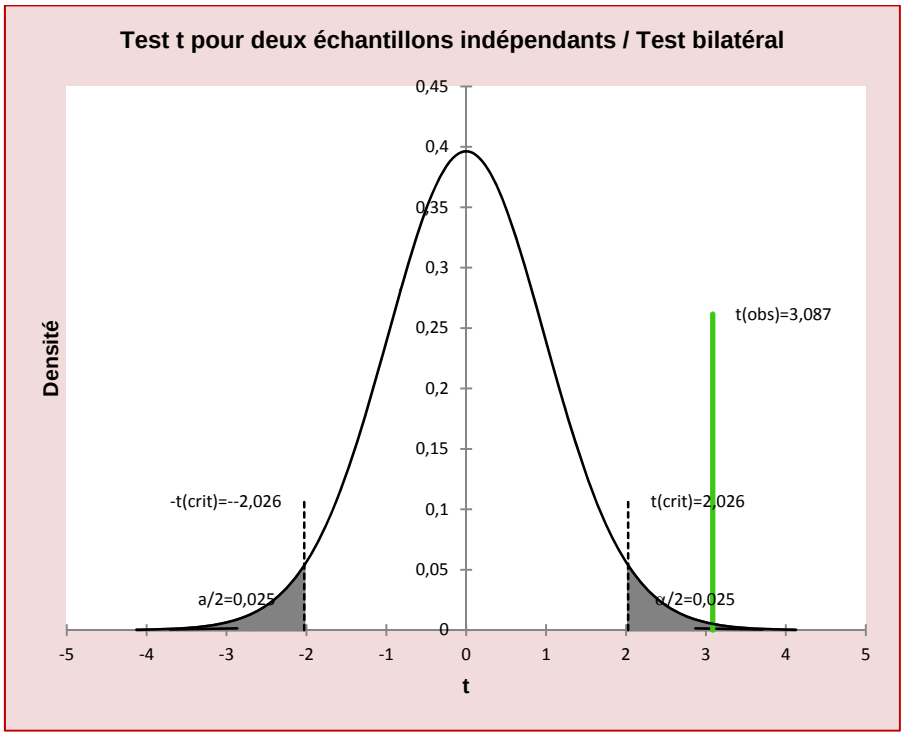



*Figure 25 : graphique du test bilatéral au post-test (test t)*

Le test de Student préconise le rejet de l'hypothèse nulle $H_0$. Il suggère donc de retenir l'hypothèse alternative. On peut donc dire que ***les moyennes des groupes sont différentes.***

D'où l'équation : $\overline{Moy}$ **GE –** $\overline{Moy}$ **GT** $\neq 0 \Rightarrow$ $\overline{Moy}$ **GE** $\neq$ $\overline{Moy}$ **GT.**

Observons maintenant les statistiques descriptives pour identifier le groupe dont la moyenne est la plus élevées.

**Tableau des résultats**

| | Post test |
|---|---|
| **GE** | 47,25 |
| **GT** | 33,84 |

*Tableau 14 : tableau des résultats au post-test*

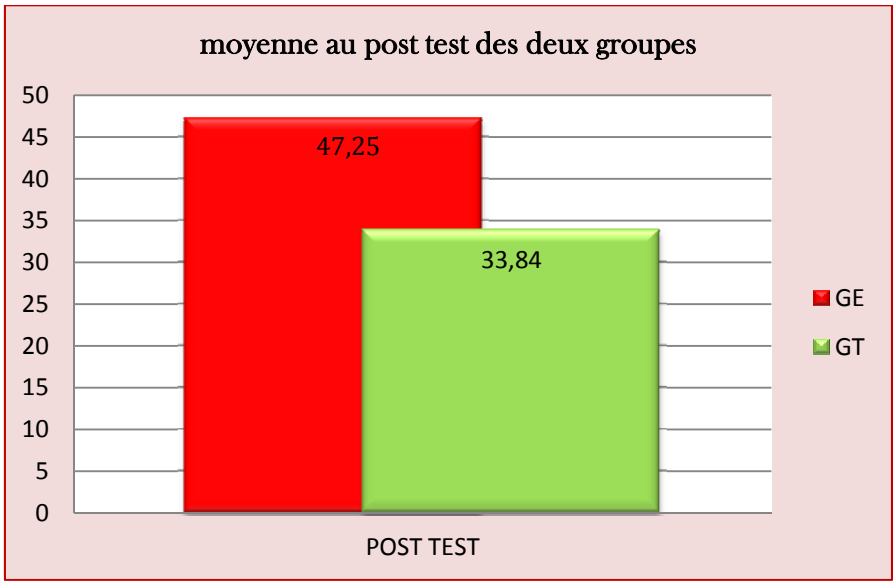



*Figure 26 : graphique des résultats au post-test*

Nous avons expliqué plus haut que si la performance au post test du groupe ayant utilisé AlgoBox est plus élevée que celle du groupe qui ne l'a pas utilisé, on peut conclure que cette application facilite l'apprentissage de l'algorithmique.

En examinant les moyennes des deux groupes, nous avons la moyenne au post test du groupe expérimental qui est de quarante-sept virgule vingt-cinq (47,25) et celle au post test du groupe témoin est de trente-trois virgule quatre-vingt-quatre (33,84).

L'équation précédente donnera :

$\overline{Moy}$ **GE –** $\overline{Moy}$ **GT** $\neq 0 \Rightarrow$ **47,25-33,84** $\neq$**0,** alors **13,14** $\neq$**0** ce qui est vrai.

De plus $\overline{Moy}$ **GE –** $\overline{Moy}$ **GT = 13,14 >0**

On a alors $\overline{Moy}$ **GE –** $\overline{Moy}$ **GT > 0** $\Rightarrow$ $\overline{Moy}$ **GE >** $\overline{Moy}$ **GT**

Nous constatons que le groupe expérimental (GE) au post test a une meilleure moyenne, quarante-sept virgule vingt-cinq (47, 25) que le groupe témoin (GT) soit trente-trois virgule quatre-vingt-quatre (33,84). La performance du groupe expérimental est supérieure d'un peu plus de treize (13) points par rapport à celle du

groupe témoin (13,41). Cet écart nous permet de conclure que la moyenne du groupe expérimentale (GE) est supérieure à celle du groupe témoin (GT).

Nous avons donc grâce au test de Student montré que la moyenne du groupe expérimental est différente de celle du groupe témoin. De plus la lecture du tableau des statistiques descriptives a mis en évidence le fait que la moyenne du groupe expérimental est supérieure à celle du groupe témoin. Ces deux éléments nous permettent de vérifier notre hypothèse. Nous pouvons donc affirmer qu'AlgoBox peut ***faciliter l'apprentissage de l'algorithmique***.

Selon notre cadre théorique, cette facilitation se fait en ***soulageant en partie*** l'apprenant ***de l'activité liée à l'apprentissage du formalisme*** par réduction de la charge cognitive intrinsèque (Paas, 1994; Sweller, 2010).

Dans cette seconde affirmation contenue dans notre hypothèse, il sera question ici de voir dans quelle mesure AlgoBox prend en charge l'apprentissage du formalisme.

Lors de l'analyse et la mise en place de notre outil de recueil de données, nous avons clairement identifié deux types de savoirs : les implicites et les explicites. Si les savoirs implicites sont liés aux connaissances acquises antérieurement par l'apprenant, les savoirs explicites émanent quant à eux directement des notions d'algorithmiques acquises en cours. Ce sont donc ces notions acquises pendant le cours d'algorithmique qui sont des connaissances déclaratives que l'apprenant doit associer à des connaissances procédurales (savoir-faire) équivalents. Ces connaissances procédurales (savoir-faire) sont des règles syntaxiques précises et rigoureuses qu'il faut respecter afin d'écrire un algorithme syntaxiquement correct. L'apprentissage du formalisme est donc lié à la maitrise de ces règles syntaxiques, ces procédures.

Concernant les savoirs implicites, il apparait aussi que les connaissances liées à ces savoirs et qui sont sollicités par le problème qui est posé à l'apprenant n'émanent pas directement du cours d'algorithmique. Les connaissances liées à ces savoirs doivent être associées à des connaissances procédurales, des savoir-faire qui sont quant à eux des règles syntaxiques apprises en cours d'algorithmique et dont la mise en œuvre représente l'apprentissage du formalisme.

En définitive, nous voyons donc que le formalisme est lié aux connaissances procédurales (savoir-faire explicites et savoir-faire implicites) en ce sens que le formalisme en algorithmique est le respect des règles syntaxiques, ces règles qui sont mis en œuvre dans l'écriture d'un algorithme par un ensemble de connaissances procédurales (savoir-faire explicites). Ainsi, l'apprentissage du formalisme est l'exercice ou la pratique de ces procédures (savoir-faire).

De plus, si nous examinons les résultats du groupe expérimental (GE) entre le pré test et le post test, les résultats concernant les connaissances liées aux savoirs et savoir-faire explicites, ces résultats peuvent également nous renseigner sur l'apport de l'application

sur les connaissances liées aux savoirs et savoir-faire explicites. En effet, en comparant la performance entre le pré test et le post test de ce groupe, nous allons voir l'apport d'AlgoBox sur les connaissances liées aux savoirs et savoir-faire explicites. Nous avons établi plus haut que les connaissances procédurales liées aux savoir-faire découlent des connaissances déclaratives liées aux savoirs et que l'apprentissage du formalisme n'est rien d'autre que celui des procédures (savoir-faire). Ainsi, si une influence contrôlée comme l'utilisation d'un logiciel augmente les résultats d'un groupe dans les connaissances liées aux savoirs et savoir-faire explicites donc cette influence facilite l'apprentissage de l'algorithmique en soulageant en partie l'apprenant de l'activité liée à l'apprentissage du formalisme.

En résumé, selon notre dispositif expérimental, nous comparons :

- au post test les performances au niveau des connaissances liées aux savoir-faire implicites et explicites du groupe ayant effectué des activités avec AlgoBox (GE) et du groupe témoin (GT) qui n'a pas utilisé l'application ;

- entre le pré test et le post test les performances du groupe expérimental ayant utilisé AlgoBox concernant les connaissances liées aux savoirs et savoir-faire explicites, nous pouvons donc voir si cette application a eu un effet sur l'apprentissage des procédures (savoir-faire).

Si le groupe expérimental (GE) par rapport au groupe témoin (GT) a de meilleurs résultats, dans le premier cas, et une augmentation de la performance du groupe expérimental (GE) entre les deux évaluations, dans la seconde situation, on peut dire que le logiciel AlgoBox améliore l'apprentissage des connaissances procédurales liées aux savoir-faire, donc qu'il soulage en partie l'apprenant de l'activité liée à l'apprentissage du formalisme.

Comme établi plus haut, il s'agira de montrer grâce au test t de Student que les groupes sont différents, puis la comparaison des moyennes identifiera le groupe qui a la plus forte moyenne.

**Test t de Student pour les savoir-faire explicites pour le GE et le GT**

- **Statistiques descriptives :**

| Variable | Observations | Obs. avec données manquantes | Obs. sans données manquantes | Minimum | Maximum | Moyenne | Écart-type |
|---|---|---|---|---|---|---|---|
| GE | 20 | 0 | 20 | 5,000 | 22,000 | 13,650 | 5,234 |
| GT | 19 | 0 | 19 | 1,000 | 21,000 | 8,789 | 5,544 |

*Tableau 15 : tableau de statistiques descriptives pour les savoir-faire explicites au post-test (test t)*

- **Test t pour deux échantillons indépendants / Test bilatéral :**

Intervalle de confiance à 95% autour de la différence des moyennes : [1,364; 8,357[

| | |
|---|---|
| Différence | 4,861 |
| t (Valeur observée) | 2,816 |
| \|t\| (Valeur critique) | 2,026 |
| DDL | 37 |
| p-value (bilatérale) | 0,008 |
| Alpha | 0,05 |

*Tableau 16 : tableau de statistiques du test bilatéral pour les savoir-faire explicites au post-test (test t)*

- **Interprétation du test :**

$H_0$ : La différence entre les moyennes est égale à 0.
$H_a$ : La différence entre les moyennes est différente de 0.
Étant donné que la p-value calculée est inférieure au niveau de signification alpha=0,05comme l'illustre le tableau 16 et la figure 27, on doit *rejeter l'hypothèse nulle $H_0$, et retenir l'hypothèse alternative $H_a$.*
Le risque de rejeter l'hypothèse nulle $H_0$ alors qu'elle est vraie est inférieur à 0,77%.

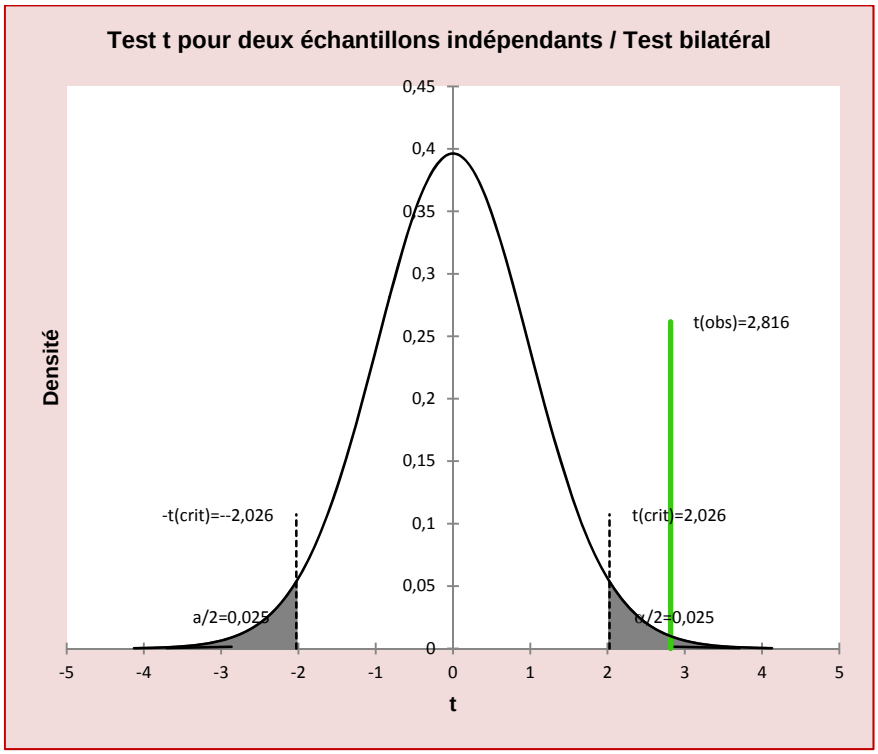



*Figure 27 : graphique du test bilatéral pour les savoir-faire explicites au post-test (test t)*

Pour les savoir-faire explicites, le test de Student préconise le rejet de l'hypothèse nulle $H_0$. Il suggère donc de retenir l'hypothèse alternative. On peut donc dire que ***les moyennes des groupes sont différentes.***

D'où l'équation : $\overline{Moy}$ GE – $\overline{Moy}$ GT ≠ 0 ⇒ $\overline{Moy}$ GE ≠ $\overline{Moy}$ GT.

**Test t de Student pour les savoir-faire implicites pour le GE et le GT**

- **Statistiques descriptives :**

| Variable | Observations | Obs. avec données manquantes | Obs. sans données manquantes | Minimum | Maximum | Moyenne | Écart-type |
|---|---|---|---|---|---|---|---|
| GE | 20 | 0 | 20 | 0,000 | 8,000 | 3,650 | 2,300 |
| GT | 19 | 0 | 19 | 0,000 | 6,000 | 2,263 | 1,881 |

*Tableau 17 : tableau de statistiques descriptives pour les savoir-faire implicites au post-test (test t)*

- **Test t pour deux échantillons indépendants / Test bilatéral :**

Intervalle de confiance à 95% autour de la différence des moyennes : [0,019; 2,754[

| | |
|---|---|
| Différence | 1,387 |
| t (Valeur observée) | 2,055 |
| \|t\| (Valeur critique) | 2,026 |
| DDL | 37 |
| p-value (bilatérale) | 0,047 |
| Alpha | 0,05 |

*Tableau 18 : tableau de statistiques du test bilatéral pour les savoir-faire implicites au post test (test t)*

- **Interprétation du test :**

$H_0$ : La différence entre les moyennes est égale à 0.
$H_a$ : La différence entre les moyennes est différente de 0.
Étant donné que la p-value calculée est inférieure au niveau de signification alpha=0,05 que l'on observe au tableau 18 et à la figure 28, on doit *rejeter l'hypothèse nulle $H_0$, et retenir l'hypothèse alternative $H_a$.*
Le risque de rejeter l'hypothèse nulle $H_0$ alors qu'elle est vraie est inférieur à 4,70%.

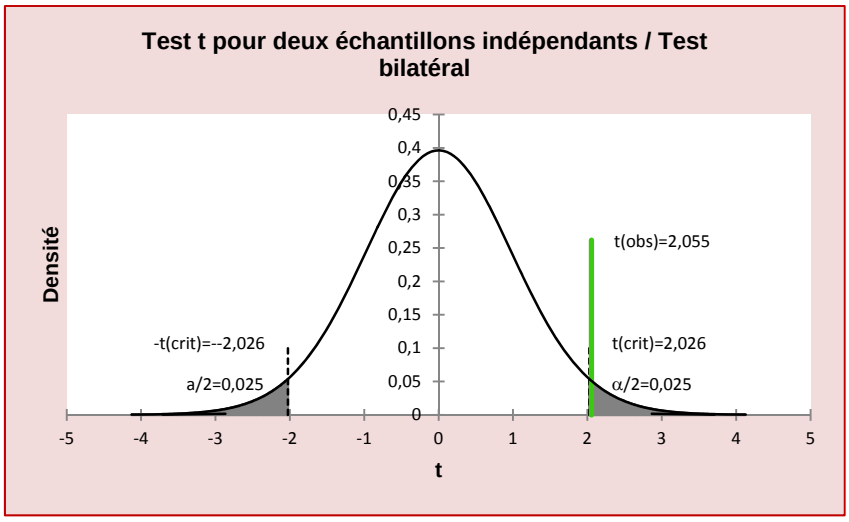



*Figure 28 : graphique du test bilatéral pour les savoir-faire implicites au post-test (test t)*

Pour les savoir-faire implicites, le test de Student préconise le rejet de l'hypothèse nulle $H_0$. Il suggère donc de retenir l'hypothèse alternative. On peut donc dire que ***les moyennes des groupes sont différentes.***

D'où l'équation : $\overline{Moy}$ **GE –** $\overline{Moy}$ **GT** $\neq 0 \Rightarrow$ $\overline{Moy}$ **GE** $\neq$ $\overline{Moy}$ **GT.**

**Tableau de résultats.**

**a- savoir-faire explicites.**

| | Post test |
|---|---|
| **GE** | 13,65 |
| **GT** | 8,79 |

*Tableau 19 : tableau des résultats au post-test pour les savoir-faire explicites.*

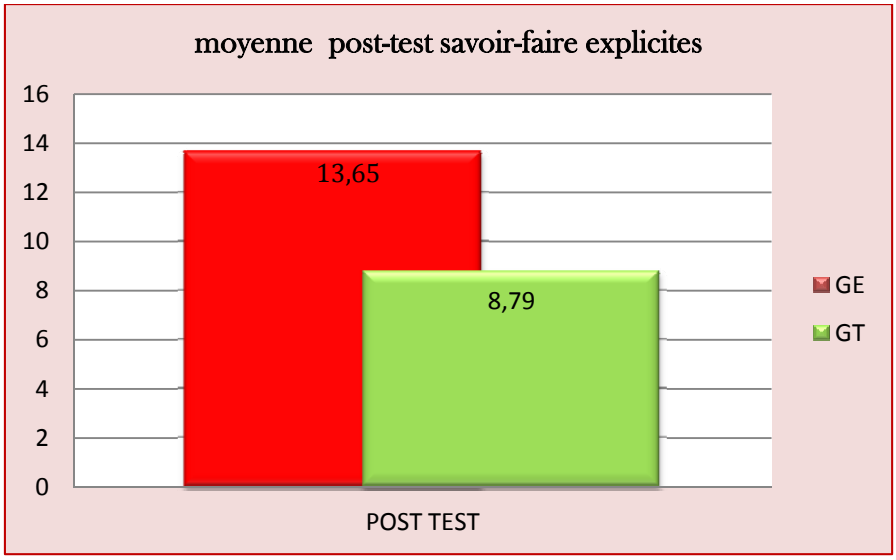



*Figure 29 : graphique des résultats au post-test pour les savoir-faire explicites.*

L'équation précédente donnera :

$\overline{Moy}$ **GE –** $\overline{Moy}$ **GT** $\neq 0 \Rightarrow$ **13,65-08,79** $\neq$**0,** alors **04,86** $\neq$**0** ce qui est vrai.

De plus $\overline{Moy}$ **GE –** $\overline{Moy}$ **GT = 04,86 >0**

On a alors $\overline{Moy}$ **GE –** $\overline{Moy}$ **GT > 0** $\Rightarrow$ $\overline{Moy}$ **GE >** $\overline{Moy}$ **GT**

**b- savoir-faire implicites**

| | Post test |
|---|---|
| **GE** | 3,65 |
| **GT** | 2,26 |

*Tableau 20 : tableau des résultats au post-test pour les savoir-faire implicites*

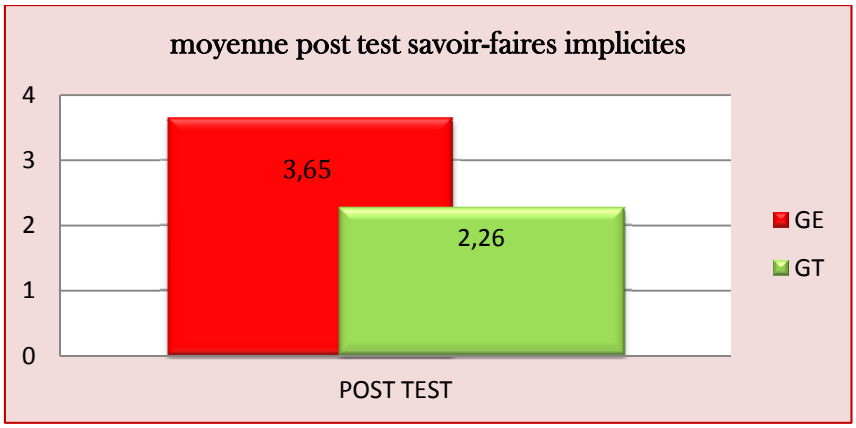



*Figure 30 : graphique des résultats au post test pour les savoir-faire implicites*

L'équation précédente donnera :

$\overline{Moy}$ **GE –** $\overline{Moy}$ **GT ≠ 0 ⇒ 03,65-02,26 ≠0,** alors **01,36 ≠0** ce qui est vrai.

De plus $\overline{Moy}$ **GE –** $\overline{Moy}$ **GT = 01,36 >0**

On a alors $\overline{Moy}$ **GE –** $\overline{Moy}$ **GT > 0 ⇒** $\overline{Moy}$ **GE >** $\overline{Moy}$ **GT**

**Test t de Student pour les savoirs et savoir-faire explicites du GE entre le pré et post test**

- **Statistiques descriptives :**

| Variable | Observations | Obs. avec donnée manquantes | Obs. sans données manquantes | Minimum | Maximum | Moyenne | Écart-type |
|---|---|---|---|---|---|---|---|
| Pré test | 40 | 0 | 40 | 0,000 | 20,000 | 8,450 | 6,496 |
| Post test | 40 | 0 | 40 | 5,000 | 25,000 | 18,075 | 6,091 |

*Tableau 21 : tableau de statistiques descriptives au pré et post test pour les savoirs et savoir-faire explicites du GE*

- **Test t pour deux échantillons indépendants / Test bilatéral :**

Intervalle de confiance à 95% autour de la différence des moyennes : [-12,428 ; -6,822[

| | |
|---|---|
| Différence | -9,625 |
| t (Valeur observée) | -6,836 |
| \|t\| (Valeur critique) | 1,991 |
| DDL | 78 |
| p-value (bilatérale) | < 0,0001 |
| alpha | 0,05 |

*Tableau 22 : tableau de statistiques du test bilatéral au pré et post-test pour les savoirs et savoir-faire explicites du GE*

- **Interprétation du test :**

$H_0$ : La différence entre les moyennes est égale à 0.
$H_a$ : La différence entre les moyennes est différente de 0.
Étant donné que la p-value calculée est inférieure au niveau de signification alpha=0,05 comme présenté au tableau 22 et à la figure 31, on *doit donc rejeter l'hypothèse nulle $H_0$, et retenir l'hypothèse alternative $H_a$.*
Le risque de rejeter l'hypothèse nulle $H_0$ alors qu'elle est vraie est inférieur à 0,01%.

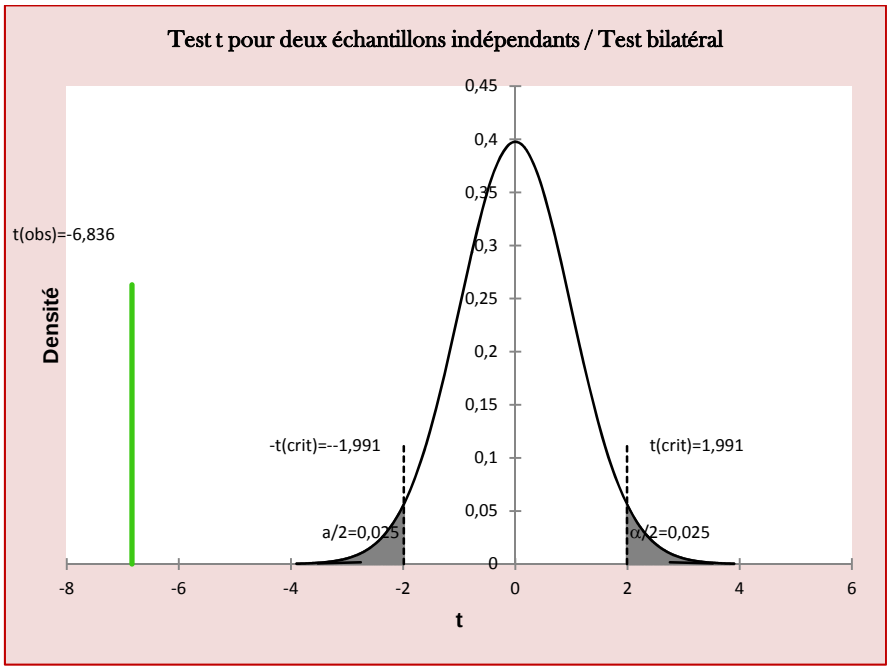



*Figure 31 : graphique du test bilatéral au pré et post-test pour les savoirs et savoir-faire explicites du GE*

Pour les savoir-faire explicites du GE, le test de Student préconise le rejet de l'hypothèse nulle $H_0$. Il suggère donc de retenir l'hypothèse alternative. On peut donc dire que ***les moyennes des groupes sont différentes.***

D'où l'équation: $\overline{Moy}$ **Post test –** $\overline{Moy}$ **Pré-test** $\neq 0 \Rightarrow$ $\overline{Moy}$ **Post test** $\neq$ $\overline{Moy}$ **Pré-test.**

**Tableau des résultats**

**a- savoirs et savoir-faire explicites (groupe expérimental)**

| | **GE** |
|---|---|
| **Pré test** | 8,45 |
| **Post test** | 18,08 |

*Tableau 23 : tableau des résultats du GE au pré et post-test pour les savoirs et savoir-faire explicites*

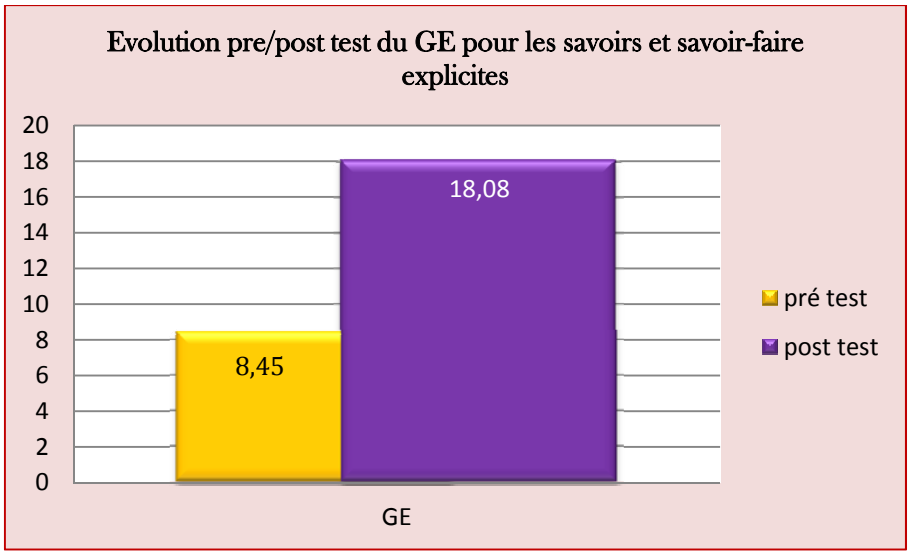



*Figure 32 : Graphique des résultats du GE au pré et post-test pour les savoirs et savoir-faire explicites*

L'équation précédente donnera :

$\overline{Moy}$ **Post test –** $\overline{Moy}$ **Pré-test ≠ 0 ⇒ 18,08-08,45 ≠0,** alors **09,63 ≠0** ce qui est vrai.

De plus $\overline{Moy}$ **Post test –** $\overline{Moy}$ **Pré-test = 09,63 >0**

On a alors $\overline{Moy}$ **Post test –** $\overline{Moy}$ **Pré-test > 0 ⇒** $\overline{Moy}$ **Post test >** $\overline{Moy}$ **Pré-test**

### 3.2.4.3. Vérification de la deuxième hypothèse

Notre seconde hypothèse stipule que : « l'élaboration d'un algorithme sollicite chez un apprenant l'accomplissement de deux tâches, l'une est la résolution du problème et l'autre est la gestion du formalisme de l'algorithme. Ainsi, la réactivation des savoirs implicites portés par la tâche prescrite, et qui seront sollicités dans l'élaboration de la solution, contribue à faciliter la compréhension et la résolution du problème, tandis qu'AlgoBox va soulager l'apprenant dans la gestion du formalisme de l'algorithme. »

Pour tester cette hypothèse, nous avons dans le but de réactiver les connaissances liées aux savoirs implicites portées par la tâche prescrite, comme le demande notre hypothèse, ajouté à nos exercices (tâche prescrite) des indications (rappel sur la notion sollicitée) afin de réactiver ou d'énoncer (pour ceux qui ne la connaissent pas) la connaissance liée aux savoirs implicites. Cette réactivation s'est faite dans l'élaboration des exercices du post test.

En guise d'exemple, dans l'exercice qui consistait à écrire un algorithme qui lit un nombre entier et vous affiche après vérification si ce nombre est pair ou impair, la réactivation ou l'énoncé des connaissances liées aux savoirs implicites portées par la tâche prescrite a consisté ici à rappeler aux étudiants qu'un nombre N est pair si et seulement si N **mod** 2 est égal à 0 (**mod** étant l'opérateur arithmétique modulo qui donne le reste de la division de deux entiers).

Nous pouvons donc observer l'effet de cette réactivation en examinant le résultat au post test (car les indications sur les connaissances implicites ont été données dans les énoncés du post test) de nos sujets qui l'ont passé, à savoir le groupe expérimental (GE) et le groupe témoin (GT). Étant donné que les mêmes sujets ont passé le pré test et que lors du pré test nous n'avons pas donné les indications qui permettent l'activation ou la réactivation des connaissances liées aux savoirs implicites. Les résultats au pré test des deux groupes constitueront donc le second élément de comparaison (c'est la situation avant l'activation des connaissances). Ce pendant, il est important de rappeler que l'activation concerne les savoirs implicites donc des connaissances acquises antérieurement pas pendant les enseignements d'algorithmique. Mais les connaissances déclaratives liées à ce savoir seront mis en œuvre en les traduisant en connaissances procédurales liées aux savoir-faire appris au cours des enseignements d'algorithmique. Ainsi l'activation des connaissances déclaratives liées aux savoirs implicites dans la tâche prescrite sera vérifiée dans la réalisation de ladite tâche prescrite. Cette réalisation consiste à transformer les connaissances déclaratives liées aux savoirs implicites réactivés en connaissances procédurales liées aux savoir-faire implicites donc en ligne d'algorithme. Finalement, tester cette hypothèse consistera à comparer les résultats des connaissances liées aux savoirs et savoir-faire implicites des deux groupes entre le pré test et le post test.

Cependant, nous pouvons aussi faire remarquer que cette amélioration de performances au post test des deux groupes peut être due non seulement à la réactivation des connaissances liées aux savoirs implicites portées par la tâche prescrite, mais aussi au fait que l'un des deux groupes, le groupe expérimental (GE), a utilisé l'application AlgoBox avant de passer le post test. Donc les performances de ce groupe sont aussi liées à ce facteur. Il nous parait donc plus judicieux pour analyser uniquement l'apport de la réactivation ou l'activation de connaissances, d'examiner la performance du groupe dont le seul facteur provoqué dans l'expérimentation a été cette réactivation ou activation de connaissances. De ce fait, nous savons qu'après le pré test, le groupe témoin (GT) a passé le post test suite à la réactivation de connaissances liées aux savoirs implicites portées par la tâche prescrite sans un autre facteur provoqué par l'expérimentation. La comparaison des performances de ce groupe entre le pré test et le post test nous renseigne mieux sur l'effet de cette réactivation ou activation de connaissances liées aux savoirs implicites portés par la tâche prescrite.

Finalement, cette hypothèse sera vérifiée en comparant les performances du groupe témoin (GT) entre le pré test et le post test. Cette comparaison sera effectuée pour les connaissances liées aux savoirs et savoir-faire implicites comme l'indique l'hypothèse.

Comme nous l'avons fait pour la précédente hypothèse, le test de Student nous permettra de démontrer que les moyennes au pré test et au post test du groupe témoin sont différentes et que l'hypothèse alternative est retenue. Le tableau des statistiques descriptives nous permettra de comparer les deux moyennes. Ainsi, si la moyenne au

post test est supérieure à celle du pré test, on dira qu'il y a amélioration des performances entre le pré test et le post test.

**Savoirs implicites**

**a-Test de Student**

- **Statistiques descriptives :**

| Variable | Observations | Obs. avec données manquantes | Obs. sans données manquantes | Minimum | Maximum | Moyenne | Écart-type |
|---|---|---|---|---|---|---|---|
| Post test | 19 | 0 | 19 | 1,000 | 9,000 | 5,947 | 3,135 |
| Pré test | 19 | 0 | 19 | 0,000 | 7,000 | 3,053 | 2,614 |

*Tableau 24 : tableau de statistiques descriptives des savoirs implicites du GT*

- **Test t pour deux échantillons indépendants / Test bilatéral :**

Intervalle de confiance à 95% autour de la différence des moyennes : [0,996; 4,794[

| | |
|---|---|
| Différence | 2,895 |
| t (Valeur observée) | 3,091 |
| \|t\| (Valeur critique) | 2,028 |
| DDL | 36 |
| p-value (bilatérale) | 0,004 |
| alpha | 0,05 |

*Tableau 25 : tableau de statistiques du test bilatéral du GT pour les savoirs implicites*

- **Interprétation du test :**

$H_0$ : La différence entre les moyennes est égale à 0.
$H_a$ : La différence entre les moyennes est différente de 0.
Étant donné que la p-value calculée est inférieure au niveau de signification alpha=0,05 comme illustré au tableau 25 et à la figure 33, on doit donc *rejeter l'hypothèse nulle* $H_0$, et *retenir l'hypothèse alternative* $H_a$.
Le risque de rejeter l'hypothèse nulle $H_0$ alors qu'elle est vraie est inférieur à 0,38%.

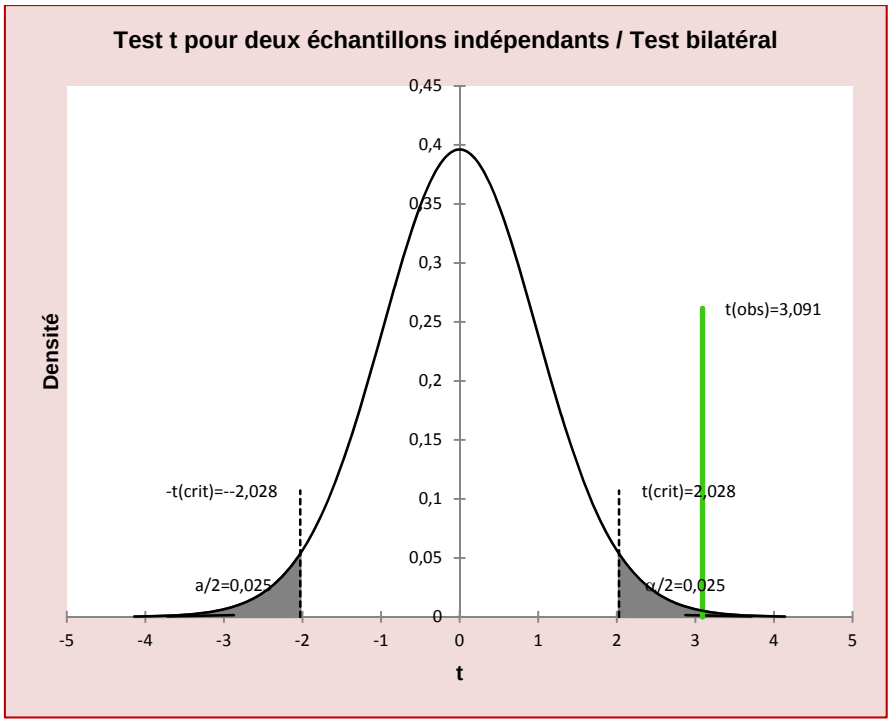



*Figure 33 : graphique du test bilatéral du GT pour les savoirs implicites.*

Pour les savoirs implicites du GT, le test de Student préconise le rejet de l'hypothèse nulle $H_0$. Il suggère donc de retenir l'hypothèse alternative. On peut donc dire que ***les moyennes des groupes sont différentes.***

On peut donc déduire l'équation :

$$\overline{Moy}\ \textbf{Post test} - \overline{Moy}\ \textbf{Pré-test} \neq 0 \Rightarrow \overline{Moy}\ \textbf{Post test} \neq \overline{Moy}\ \textbf{Pré-test.}$$

## b- Tableau des résultats

| | GT |
|---|---|
| **Pré test** | 3,05 |
| **Post test** | 5,95 |

*Tableau 26 : tableau des résultats du GT pour les savoirs implicites*

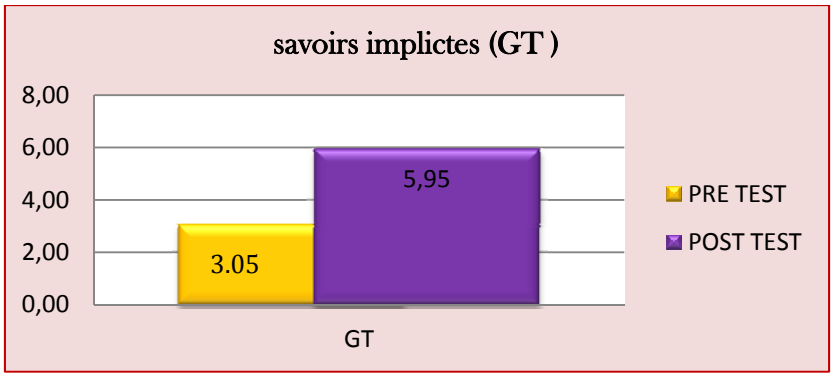



*Figure 34 : graphique des résultats du GT pour les savoirs implicites*

L'équation précédente donnera :

**$\overline{Moy}$ Post test – $\overline{Moy}$ Pré-test ≠ 0 ⇒ 05,95-03,05 ≠0,** alors **02,90 ≠0** ce qui est vrai.

De plus **$\overline{Moy}$ Post test – $\overline{Moy}$ Pré-test = 02,90 >0**

On a alors **$\overline{Moy}$ Post test – $\overline{Moy}$ Pré-test > 0 ⇒ $\overline{Moy}$ Post test > $\overline{Moy}$ Pré-test**

**Savoir-faire implicites**

**a- Test de Student**

- **Statistiques descriptives :**

| Variable | Observations | Obs. avec données manquantes | Obs. sans données manquantes | Minimum | Maximum | Moyenne | Écart-type |
|---|---|---|---|---|---|---|---|
| Post test | 19 | 0 | 19 | 0,000 | 6,000 | 2,263 | 1,881 |
| Pré test | 19 | 0 | 19 | 0,000 | 4,000 | 0,263 | 0,933 |

*Tableau 27 : tableau de statistiques descriptives du GT pour les savoir-faire implicites*

- **Test t pour deux échantillons indépendants / Test bilatéral :**

Intervalle de confiance à 95% autour de la différence des moyennes : [1,023; 2,977[

| | |
|---|---|
| Différence | 2,000 |
| t (Valeur observée) | 4,152 |
| \|t\| (Valeur critique) | 2,028 |
| DDL | 36 |
| p-value (bilatérale) | 0,000 |
| alpha | 0,05 |

*Tableau 28 : tableau de statistiques du test bilatéral du GT pour les savoir-faire implicites*

- **Interprétation du test :**

$H_0$ : La différence entre les moyennes est égale à 0.
$H_a$ : La différence entre les moyennes est différente de 0.
Étant donné que la p-value calculée est inférieure au niveau de signification alpha=0,05 comme présenté sur le tableau 29 et à la figure 35, on doit *rejeter l'hypothèse nulle $H_0$,* et *retenir l'hypothèse alternative $H_a$.*
Le risque de rejeter l'hypothèse nulle $H_0$ alors qu'elle est vraie est inférieur à 0,02%.

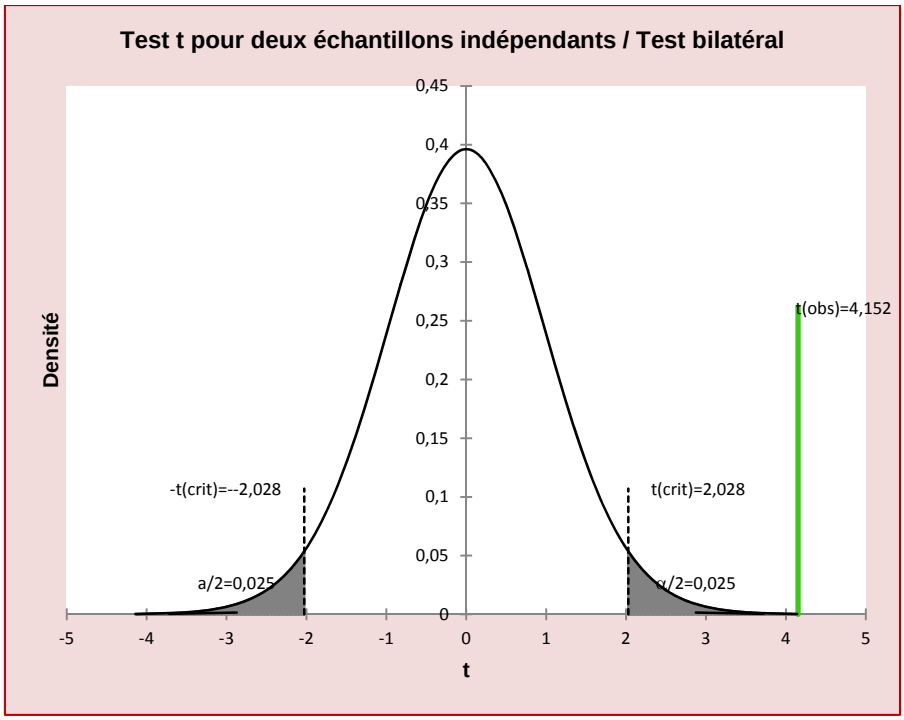



*Figure 35 : graphique du test bilatéral du GT pour les savoir-faire implicites.*

Pour les savoir-faire implicites du GT, le test de Student préconise le rejet de l'hypothèse nulle $H_0$. Il suggère donc de retenir l'hypothèse alternative. On peut donc dire que ***les moyennes des groupes sont différentes.***

D'où l'équation: $\overline{Moy}$ **Post test –** $\overline{Moy}$ **Pré-test** $\neq 0 \Rightarrow$ $\overline{Moy}$ **Post test** $\neq$ $\overline{Moy}$ **Pré-test.**

## b- Tableau des résultats

| | **GT** |
|---|---|
| **Pré test** | 0,26 |
| **Post test** | 2,26 |

*Tableau 29 : tableau des résultats du GT pour les savoir-faire implicites*

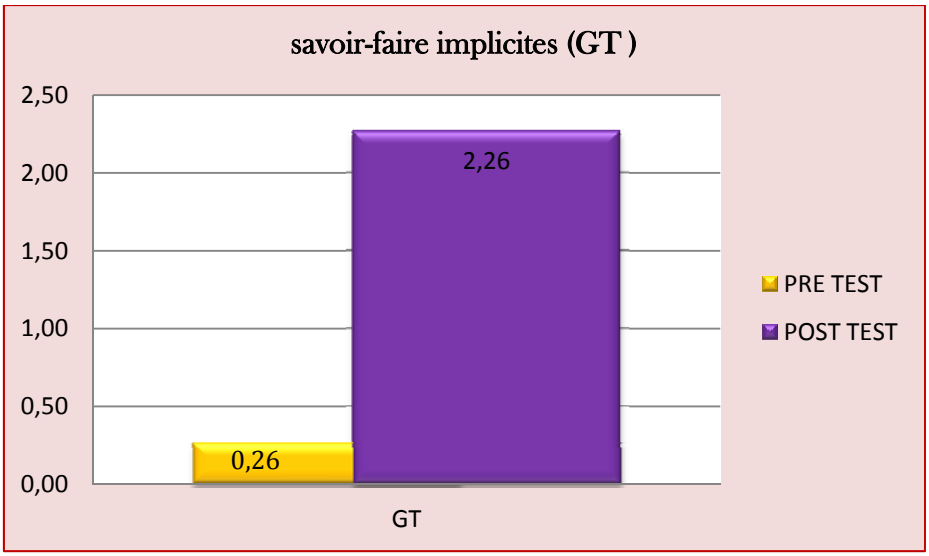



*Figure 36 : graphique des résultats du GT pour les savoir-faire implicites*

L'équation précédente donnera :

$\overline{Moy}$ **Post test –** $\overline{Moy}$ **Pré-test ≠ 0 ⇒ 02,26-0,26 ≠0,** alors **02,00 ≠0** ce qui est vrai.

De plus $\overline{Moy}$ **Post test –** $\overline{Moy}$ **Pré-test = 02,00 >0**

On a alors $\overline{Moy}$ **Post test –** $\overline{Moy}$ **Pré-test > 0 ⇒** $\overline{Moy}$ **Post test >** $\overline{Moy}$ **Pré-test**

# 3.3. Discussions

## 3.3.1. Discussion sur la première expérimentation

De notre première expérimentation, il se dégage donc que l'apprenant est confronté à deux principales difficultés lors de la résolution d'un exercice:

- l'une liée à la compréhension d'une notion de cours (savoir et les connaissances déclaratives liées à ce savoir) et à la mise en œuvre de cette notion de manière pratique (connaissances procédurales liées au savoir-faire) lors de la résolution d'un exercice ;

- l'autre liée à la maitrise d'une connaissance liée à un savoir implicite indispensable pour résoudre un exercice. Il est donc important de s'interroger sur la manière de résoudre ces difficultés.

Dans le premier cas, amener l'apprenant à apprendre des savoirs afin que ces savoirs soient utilisés lors de la résolution d'exercices signifie que l'apprenant va apprendre d'autres connaissances par la pratique : ce sont les connaissances procédurales associées aux savoir-faire. Il a toujours en mémoire les connaissances déclaratives qui lui permettent d'interpréter la tâche et de comprendre ce qu'il doit faire. Mais ces deux types de connaissances ne sont pas les mêmes, ce qui pourrait évidemment expliquer les difficultés de résolution de problèmes en algorithmique. Cependant, selon Tardif (1992)

les connaissances procédurales permettent l'action. Le même auteur poursuit en disant que l'acquisition des connaissances procédurales se fait par la pratique des exercices faits à la suite du cours. Les exercices auraient le pouvoir de transformer des connaissances déclaratives en connaissances participant à réalisation de tâches telles qu'on les trouve dans la résolution de problèmes (Tardif, 1992). Dans notre recherche, nous complétons ces exercices par des travaux pratiques avec une application informatique de simulation d'algorithmes. Ce qui convoque ici l'un des multiples usages de l'ordinateur dans l'enseignement. Dans ce cas de figure, l'application sera utilisée pour aider à comprendre et résoudre l'exercice. On va simuler progressivement les différentes solutions proposées par l'apprenant jusqu'à l'obtention d'un programme (algorithme) qui fonctionne. Cette démarche présente l'avantage que l'apprenant apprend par essais-erreurs en manipulation, il découvre les bons codes associés aux savoirs qu'il a identifié, il simule et découvre ce que ledit code produit comme résultat. Ainsi les connaissances procédurales que l'apprenant a associées aux déclaratives sont directement testées, placées en mémoire si ça fonctionne. L'apprenant retient donc des certitudes (puisqu'il a vu que ça fonctionne), et cherche à lever les doutes quand cela ne fonctionne pas en testant d'autres procédures.

Concernant la seconde difficulté, la maitrise d'une connaissance liée à un savoir implicite, donc d'une notion qui n'est pas directement enseignée dans le cours. Mais cependant, force est de constater que si l'apprenant n'active pas la bonne connaissance liée à ce savoir, il lui est impossible de résoudre le problème comme nous l'avons déjà signalé. Il se pose donc ici un problème de réactivation de connaissances nécessaires à la mise en œuvre de l'algorithme, particulièrement de la réactivation de connaissances qui relèvent de savoirs implicites. Afin de résoudre le problème, la bonne connaissance doit être activée. Un savoir non appris, qu'il soit implicite ou explicite, sera toujours un obstacle à la mise en œuvre de la résolution d'un problème du fait de l'activation par l'apprenant de la mauvaise connaissance. De plus, en cours d'algorithmique, les savoirs que l'apprenant doit maitriser, sont ceux liés à l'algorithmique. Aussi, tous autres savoirs ne doivent être des entraves à l'écriture de l'algorithme. En d'autres termes, dans un cours d'algorithmique (informatique), l'enseignant doit évaluer non seulement la compréhension et la mise en œuvre des connaissances liées à sa discipline, mais aussi de tous les autres savoirs implicites. En définitive, pour permettre à l'apprenant de résoudre un exercice d'algorithmique, l'enseignant en situation d'enseignement-apprentissage doit effectivement prendre en compte dans son enseignement des savoirs implicites qui sont en jeu. S'il ne peu pas les revoir entièrement, il peut lors de la confection des énoncés les rappeler aux apprenants.

Au regard de ce qui précède, le recours à AlgoBox permet de soutenir l'apprentissage de l'algorithmique. Durant le déroulement du module, les travaux dirigés qui seront donnés aux apprenants seront corrigés en travaux pratiques où chacun devra encoder puis simuler sa proposition de solution jusqu'à ce qu'elle fonctionne correctement. Ainsi, les erreurs seront directement identifiées, corrigées jusqu'à la réécriture de la solution définitive. La difficulté liée à l'apprentissage des règles syntaxiques, c'est-à-dire le

formalisme sera amoindrie par la prise en charge de cette partie du travail par l'application (réduction de la charge cognitive). En effet, les codes sont directement accessibles dans l'application par un simple clic.

De plus, les énoncés des exercices donnés aux apprenants seront libellés de telle manière que les savoirs implicites sollicités par l'exercice seront clairement rappelés sous la forme d'une formule mathématique, d'une règle ou un exemple de calcul.

En résumé, cette première expérimentation nous a permis, en nous appuyant sur un certain nombre de savoirs et savoir-faire, d'identifier des indicateurs sur lesquels nous nous sommes basés pour comprendre les difficultés d'apprentissage des apprenants. Pour ce faire, nous avons confronté chaque indicateur à la production de l'apprenant afin d'identifier l'erreur commise, le type d'erreur et ce qui peut être à l'origine de cette erreur. Finalement, par rapport à l'ensemble des indicateurs, il ressort d'importantes difficultés d'apprentissage comme en témoigne les performances de l'ensemble des sujets dans l'ensemble des exercices proposés.

## 3.3.2. Discussion sur la deuxième expérimentation

### 3.3.2.1. Réponse à la Première hypothèse

En procédant à une comparaison directe des moyennes au post test concernant les connaissances liées aux savoir-faire explicites et implicites du groupe expérimental (GE) et du groupe témoin (GT), on constate des meilleurs résultats du GE par rapport au GT. En effet le GE qui a réalisé des activités avec AlgoBox obtient treize virgule soixante-cinq (13,65) aux connaissances procédurales liées aux savoir-faire explicites et trois virgule soixante-cinq (3,65) aux connaissances procédurales liées aux savoir-faire implicites contre huit virgule soixante-dix-neuf (8,79) aux connaissances procédurales liées savoir-faire explicites et deux virgule vingt-six (2,26) pour les connaissances procédurales liées aux savoir-faire implicites concernant le GT. Il est à signaler que ces moyennes sont sur vingt-cinq (25) pour les connaissances liées aux savoir-faire explicites et neuf (9) pour les connaissances liées aux savoir-faire implicites. On constate donc ici dans les deux cas de connaissances procédurales liées aux savoir-faire que la moyenne du groupe expérimental (GE) est supérieure à la moyenne du groupe témoin (GT).

Le test de Student confirme le rejet de l'hypothèse nulle ce qui implique de retenir l'hypothèse alternative, c'est-à-dire que les groupes sont différents.

Ces résultats montrent que le groupe ayant mené des activités avec AlgoBox présente de meilleures performances que celui qui ne l'a pas utilisé lorsqu'il s'est agit de tester les connaissances procédurales liées aux savoir-faire compatibles à l'apprentissage du formalisme.

De même, la comparaison des moyennes du groupe expérimental entre le pré test et le post test concernant les connaissances liées aux savoirs et savoir-faire explicites montre clairement une nette augmentation de la performance. En effet, au pré test, le

groupe expérimental a eu une moyenne de huit virgule quarante-cinq (8,45). Cette performance a été améliorée au post test passant ainsi à une moyenne de dix-huit virgule zéro huit (18,08), soit un ratio post test sur pré test de deux virgule quatorze (2,14). Donc la moyenne au post test est plus de deux fois plus grande que celle au pré test.

Nous pouvons donc dire qu'il y a facilitation de l'activité liée à l'apprentissage du formalisme.

Il était question de monter ici qu'AlgoBox facilite l'activité liée à l'apprentissage du formalisme. Nous avons donc testé les résultats concernant les connaissances procédurales liées aux savoir-faire explicites et implicites au post test du GE et du GT. De plus, notre cadre théorique a mis en exergue le fait que l'apprentissage de l'algorithmique implique deux activités cognitives : l'apprentissage du formalisme et la résolution d'un problème. Et selon Amadieu & Tricot (2006) ainsi que Sweller (2010), cette double activité ne facilite pas l'apprentissage. Ce qui signifie que les problèmes liés à l'apprentissage et les difficultés qu'ont les apprenants à comprendre cette discipline sont dus essentiellement au fait que les apprenants doivent exercer parallèlement ces deux activités mentales. Ainsi, si une partie de cette activité cognitive devient moins éprouvante, tout l'apprentissage en sera soulagé. Nous avons montré plus haut que l'activité liée au savoir-faire donc au formalisme est facilité grâce à AlgoBox ce qui signifie qu'il y a réduction de la charge cognitive de l'apprenant.

En définitive, nous démontrons ainsi qu'AlgoBox facilite l'apprentissage de l'algorithmique en soulageant l'apprenant de l'activité liée à l'apprentissage du formalisme par réduction de la charge cognitive intrinsèque.

### 3.3.2.2. Réponse à la seconde hypothèse

Lorsque nous regardons la comparaison de la moyenne du groupe témoin (GT) au pré test et au post test en ce qui concerne les savoirs et savoir-faire implicites, nous constatons une meilleure performance de ce groupe au post test, c'est-à-dire après la réactivation des connaissances liées aux savoirs implicites portées par la tâche prescrite.

Quant au test de Student, il préconise de retenir l'hypothèse alternative malgré le risque de rejeter l'hypothèse nulle $H_0$ alors qu'elle est vraie inférieur à 0,02% dans le cas des connaissances procédurales liées aux savoir-faire implicite et 0,38% dans le cas des connaissances liées aux savoirs implicites. Au regard de ce qui précède, nous pouvons donc dire que le GT a des meilleurs résultats au post test par rapport au pré test en ce qui concerne les connaissances liées aux savoirs et savoir-faire implicites.

En résumé, nous pouvons affirmer qu'une réactivation des connaissances liées aux savoirs implicites portées par la tâche prescrite, et qui seront sollicitées dans l'élaboration de la solution, facilitera chez l'apprenant la compréhension de la situation-problème.

## 3.3.3. Discussion et présentation d'autres résultats

Après ces discussions sur des situations qui nous renseignent précisément sur nos hypothèses de recherche, il nous semble important de porter un regard sur certains autres résultats qui peuvent nous fournir des informations ou ouvrir des perspectives à notre recherche.

### a-/ Les connaissances liées aux savoirs et savoir-faire implicites entre le pré et le post test pour l'échantillon.

**Tableau des résultats**

| | Échantillon. |
|---|---|
| **Pré test** | 1,94 |
| **Post test** | 4,85 |

*Tableau 30 : tableau des résultats de l'échantillon pour les savoirs et savoir-faire implicites.*

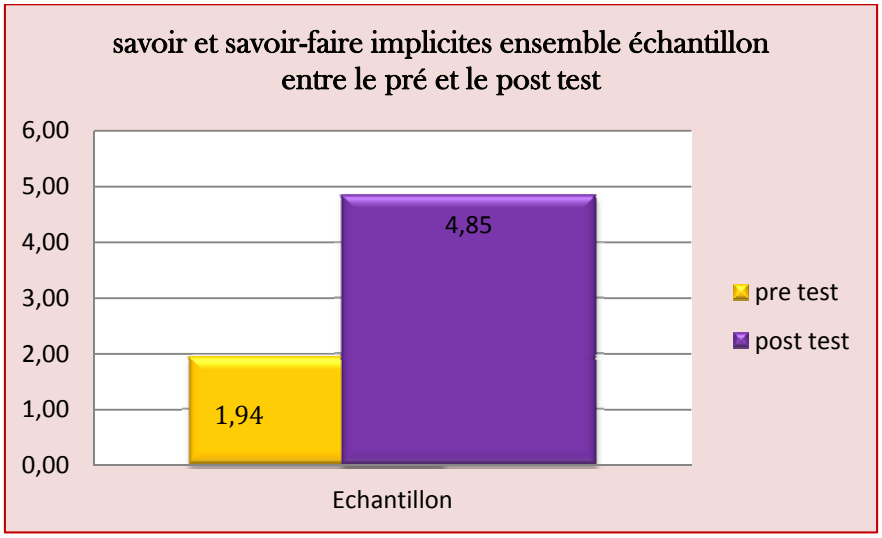



*Figure 37 : graphique des résultats de l'échantillon pour les savoirs et savoir-faire implicites.*

Lorsque nous regardons ces résultats, nous constatons que les performances des deux groupes au post test sont meilleures que celles au pré test en ce qui concerne les connaissances liées aux savoirs et savoir-faire implicites comme en témoigne l'histogramme ci-dessus. Nous savons qu'entre les deux évaluations, le groupe expérimental a subit l'influence d'AlgoBox mais aussi la réactivation des connaissances liées aux savoirs implicites. Concernant le groupe témoin, il a uniquement été assujetti à la réactivation ou l'activation des connaissances liées aux savoirs implicites. Nous pouvons donc conclure que nos sujets ont de meilleurs résultats du fait de l'influence de ces facteurs expérimentaux contrôlés. Ce résultat ne nous permet pas de conclure de manière certaine sur l'une ou l'autre de nos hypothèses, mais l'augmentation de la performance entre les deux évaluations est tout de même un élément qui oriente sur

l'influence de nos paramètres expérimentaux que nous avons introduit entre le pré test et le post test.

### b-/ Les connaissances liées aux savoirs et savoir-faire explicites entre le pré et le post test pour l'échantillon.

**Tableau des résultats**

| | Échantillon. |
|---|---|
| **Pré test** | 7,95 |
| **Post test** | 15,51 |

*Tableau 31 : tableau des résultats de l'échantillon pour les savoirs et savoir-faire explicites.*

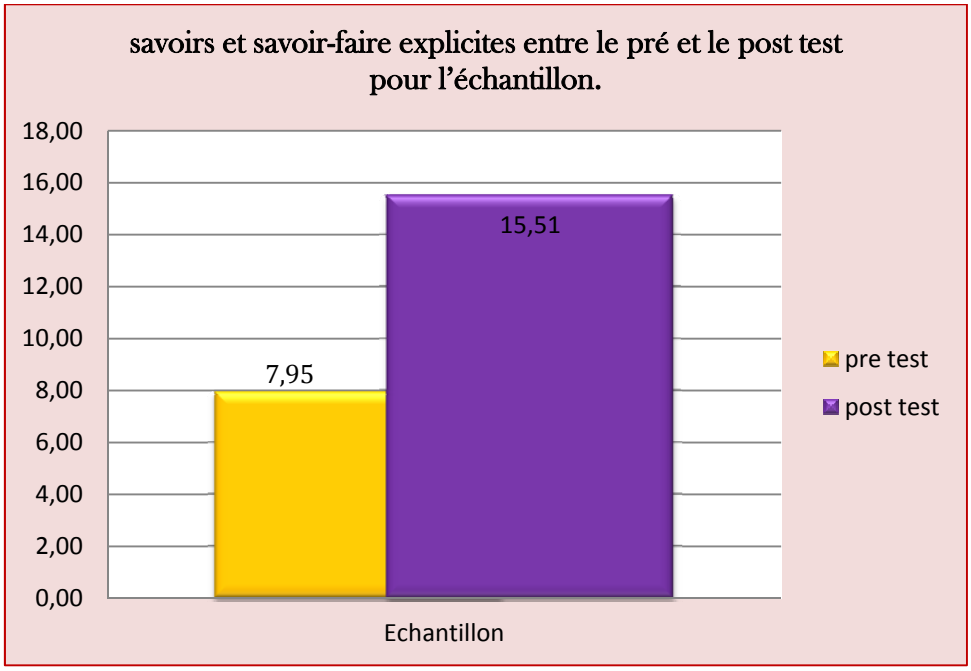



*Figure 38 : graphique des résultats de l'échantillon pour les savoirs et savoir-faire explicites*

Ces résultats montrent une moyenne de quinze virgule cinquante-un au post test (15,51) contre sept virgule quatre-vingt-quinze (7,95) au pré test pour les connaissances liées aux savoirs et savoir-faire explicites, soit un résultat au post test de presque deux fois plus élevé qu'au pré test comme illustré dans la figure 38 ci-dessus. Nous savons que dans notre échantillon, le groupe témoin n'a eu que l'influence de la réactivation des connaissances liées aux savoirs implicites. De plus, le groupe expérimental, quant à lui a été influencé par l'application et l'activation des connaissances liées aux savoirs implicites. Mais ici, il s'agit des connaissances liées aux savoirs et savoir-faire explicites. Donc l'augmentation des résultats ne peut se justifier que par l'apport d'AlgoBox au groupe expérimental. En effet, nous avons vu plus haut qu'entre les deux évaluations, pour le GE, l'écart est de presque dix points alors que pour le GT il est de cinq points environ. Il est vrai que les deux groupes améliorent leurs

performances, mais le GE, grâce à l'apport du logiciel AlgoBox, tire la performance de tout l'échantillon vers le haut.

### c-/ Les connaissances liées aux savoirs et savoir-faire explicites (groupe témoin)

### Tableau des résultats

| | GT |
|---|---|
| **Pré test** | 7,42 |
| **Post test** | 12,82 |

*Tableau 32 : tableau des résultats du GT pour les savoirs et savoir-faire explicites.*

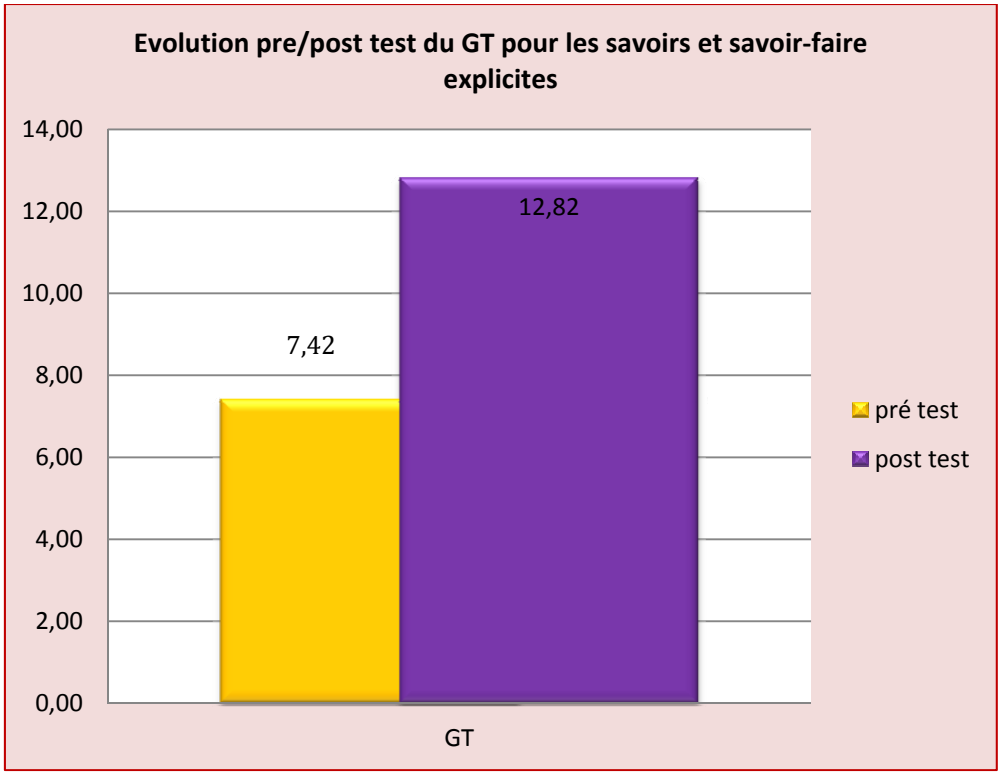



*Figure 39 : graphique des résultats du GT pour les savoirs et savoir-faire explicites.*

En observant la graphique ci-dessus, nous constatons que les performances entre le pré test et le post test du groupe témoin (GT) en ce qui concerne les connaissances liées aux savoirs et les savoir-faire explicites sont en augmentation. En effet, la moyenne au post test du groupe témoin pour ce type de connaissances liées aux savoirs et savoir-faire est de douze virgule quatre-vingt deux (12,82) contre sept virgule quarante-deux (7,42) au pré test. Il y a donc une progression de cinq virgule quatre (5,4) points de la performance du groupe témoin entre le pré test et le post test pour les connaissances liées aux savoirs et savoir-faire explicites. Nous savons que le groupe témoin pour les connaissances liées aux savoir et savoir-faire explicites n'a subit entre les deux évaluations aucune influence contrôlée émanent de notre expérimentation. Ce pendant, lors du post test, l'activation des savoirs implicites aurait-elle eut une influence sur les

connaissances liées aux savoirs et savoir-faire explicites ? De tout évidence oui, si nous nous en tenons à l'amélioration des résultats au post test donc après ladite réactivation des connaissances implicites de ce groupe. Hormis ce facteur, il ne nous reste que les paramètres non contrôlés, et le hasard qui peuvent expliquer cette augmentation de la performance entre les deux évaluations.

## 3.4. Analyse comportementale

Pendant les deux expérimentations, nous avons observé les sujets au cours de l'exécution de la tâche qui leur a été confiée. Au cours de cette observation non armée, nous avons consigné dans une grille d'observation les comportements des différents sujets pendant qu'ils travaillaient à l'élaboration d'algorithmes. Cette observation avait pour but de mettre en évidence la dimension comportementale d'introversion ou d'extraversion de notre échantillon.

D'une manière générale, il convient de rappeler qu'Eysenck (1960) a élaboré une théorie selon laquelle la personnalité serait constituée de deux facteurs principaux: le facteur N (névrosisme-stabilité) et le facteur E (introversion-extraversion). Ce dernier semble toutefois le seul dont la présence et la validité furent nettement confirmées par d'autres recherches. L'approche d'Eysenck dans le domaine de la personnalité est particulièrement intéressante dans le sens où elle relie le facteur introversion-extraversion à des aspects physiologiques du comportement et dans le sens également où elle propose que ce facteur puisse influencer bon nombre de comportements en situation expérimentale.

Au départ, il faut situer l'évolution du concept introversion-extraversion dans le contexte des recherches des facteurs constitutifs de la personnalité. A cet égard, la théorie des « traits » et des « types » connut une grande popularité auprès des chercheurs dans leur tentative d'en arriver à pouvoir identifier différentes sortes de personnalités.

Brièvement, cette théorie suggère, dans la poursuite de son objectif, les séquences suivantes : d'abord, à un premier niveau, l'individu répond de façon spécifique à différents stimuli; puis, à un deuxième niveau, ses réponses spécifiques sont regroupées en réponses habituelles; à un troisième niveau, ses réponses habituelles sont regroupées sous la notion de « traits » et enfin, à un quatrième niveau, ses traits sont regroupés sous le terme de « type ». Eysenck (1960) a souscrit à cette théorie de « traits » et de « types » développée auparavant par d'autres chercheurs et a introduit la notion de type introverti et type extraverti, ces deux types répondant à un ensemble de traits différents pour chacun d'eux. Au début, le concept introversion-extraversion fut donc considéré en fonction de la théorie des « traits » et des « types » qui reposait surtout sur l'observation de comportements et sur l'analyse factorielle. Puis, Kretschmer (1948) fit évoluer la notion de type en y introduisant une dimension physiologique. Désormais, un type de personnalité ne se résumait plus seulement à un ensemble de comportements observés mais aussi à des caractéristiques physiologiques. Eysenck (1960) poursuivit dans la même veine en reliant le concept d'introversion-extraversion à celui d'excitation-inhibition corticale.

Ainsi, l'introverti est alors considéré comme un individu ayant un fort potentiel d'excitation corticale et un faible potentiel d'inhibition corticale.

A l'inverse, l'extraverti est considéré comme un individu ayant un faible potentiel d'excitation corticale et un fort potentiel d'inhibition corticale. En d'autres termes, et toujours selon Eysenck (1960), les extravertis sont considérés comme des individus corticalement inhibés parce que, chez eux, les processus inhibitoires dominent alors que les introvertis sont considérés comme des individus corticalement excités parce que, chez eux, les processus excitatoires dominent. Et c'est au niveau du rôle de la formation réticulaire précisément que les termes « d'excitation » et « d'inhibition » corticale prennent tout leur sens. Celle-ci comprend en effet deux zones responsables de l'arrivée de stimulations au cortex. L'une, dans la partie supérieure, a pour effet « d'exciter » le cortex pour le rendre plus disponible aux informations extérieures, et l'autre, dans la partie inférieure, a pour effet de bloquer l'arrivée d'informations au cortex. En retour, ce dernier peut à son tour stimuler l'une ou l'autre des zones de la formation réticulaire dépendamment de la nature des informations transmises.

Il est constaté par les propos d'Eysenck (1967) que les concepts « d'excitation » et « d'inhibition » corticales se précisent davantage et prennent le sens « d'arousal » cortical par l'action du système d'activation réticulaire. En introduisant la notion « d'arousal » c'est-à-dire d'éveil ou d'attention corticale suscitée par l'action du système d'activation réticulaire, non seulement Eysenck faisait-il évoluer davantage le concept introversion-extraversion, mais il rejoignait en cela les propos de Gray (1967) qui avait déjà montré que le terme « arousal » s'avérait beaucoup plus fonctionnel que le terme d'excitation-inhibition corticale quand il s'agissait d'évaluer la fonction de la formation réticulaire auprès du cortex. Cette précision a permis de formuler une définition descriptive de la dimension introversion-extraversion qui a encore cours aujourd'hui: les introvertis sont caractérisés par un niveau d'éveil ou d'attention corticale plus élevé que les extravertis qui, eux, possèdent un niveau d'attention corticale relativement bas. Le concept d'introversion-extraversion délaissait donc quelque peu son aspect phénotypique pour trouver un sens plus précis au niveau physiologique et ce, par l'intermédiaire de la formation réticulaire; cela n'empêchait pas toutefois de formuler une description comportementale de l'introverti et de l'extraverti.

Les différences d'ordre physiologique entre l'introverti et l'extraverti entrainent également, au niveau phénotypique, des différences de comportement entre l'introverti et l'extraverti :

- L'extraverti typique est sociable, aime les réunions, a beaucoup d'amis, a besoin de personnes à qui parler et n'aime pas lire ou travailler tout seul. Il recherche les émotions fortes, prend des risques, fait des projets, agit sous l'impulsion du moment et est généralement un individu impulsif. Il aime beaucoup les grosses plaisanteries, a la réplique facile et aime en général le changement. Il est insouciant, peu exigeant, optimiste et aime la rigolade. Il préfère rester en mouvement et agir, a tendance à être agressif et à perdre son sang-froid rapidement. Il ne possède pas un très grand contrôle de ses sentiments et ce n'est pas toujours une personne sur qui l'on peut compter.

- L'introverti typique est le genre d'individu tranquille, effacé, introspectif, plus amateur de livres que de gens, il est réservé et distant sauf avec ses amis intimes. Il a des dispositions à la prévoyance. Il ne s'engage pas à la légère et se méfie des impulsions du moment. Il n'aime pas les sensations fortes, prend au sérieux les choses de la vie quotidienne et aime avoir une vie bien réglée. Il contrôle étroitement ses sentiments, se conduit rarement d'une manière agressive et ne s'emporte pas facilement. Il est digne de confiance, quelque peu pessimiste et accorde une grande valeur aux critères éthiques.

Pour mesurer ces différences de comportements selon l'introversion-extraversion, une grille d'analyse dont les items nous renseignent sur la nature comportementale de notre échantillon a été utilisée.

Cette observation nous a permis d'avoir les résultats suivants :

## 3.4.1. Introversion-Inhibition

### - Tableau de synthèse des résultats

| | Pas | Peu | Moyennement | Plutôt | Tout à fait |
|---|---|---|---|---|---|
| S'approprie facilement le sujet au cours de l'activité | 13 | 13 | 5 | 14 | 11 |
| Ne pose pas de question après la lecture du sujet | 6 | 6 | 7 | 19 | 18 |
| Ne consulte aucune autre source d'information | 5 | 1 | 8 | 20 | 22 |
| Se plait à soigner sa copie | 5 | 5 | 3 | 19 | 24 |
| Très lent à rentrer dans le travail | 7 | 5 | 4 | 24 | 16 |
| Reste indifférent au travail demandé | 5 | 8 | 3 | 24 | 16 |
| Attitude consentante à l'invitation à se mettre au travail | 7 | 6 | 12 | 25 | 6 |
| Mène sa tâche avec un grand souci de détail peu de choses lui échappent | 6 | 8 | 5 | 12 | 25 |
| Mène sa tâche de façon étroite, reste fixée sur le Sujet, sur le thème | 4 | 6 | 3 | 16 | 27 |
| Assume de façon régulière et jusqu'à la fin une petite responsabilité | 3 | 6 | 2 | 25 | 20 |
| parait indifférent au travail des autres | 3 | 4 | 5 | 32 | 12 |
| parait maladroit et lourd | 2 | 7 | 2 | 13 | 32 |
| montre des tendances à l'isolement | 1 | 3 | 1 | 35 | 16 |
| aborde le sujet avec calme et confiance en soi | 5 | 9 | 3 | 32 | 7 |
| commence à percevoir la nécessité d'une règle syntaxique utilisée dans la solution | 4 | 5 | 4 | 27 | 16 |
| **Total** | **76** | **92** | **67** | **337** | **268** |
| **Pourcentage** | **9%** | **11%** | **8%** | **40%** | **32%** |

*Tableau 33 : tableau de synthèse des résultats introversion-inhibition.*

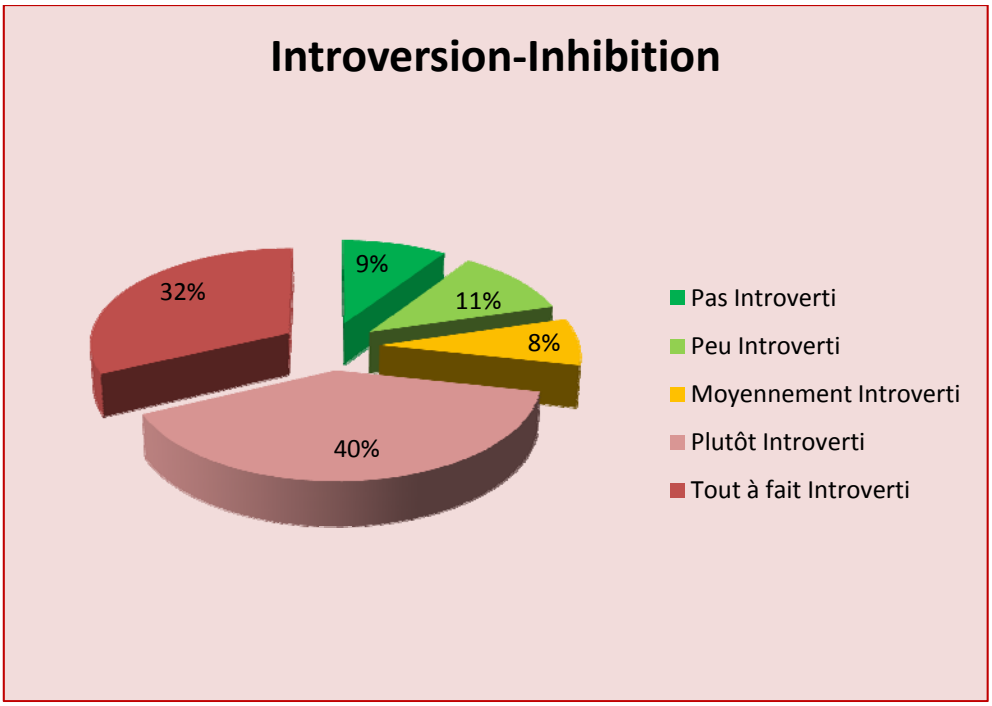



*Figure 40 : graphique synthèse des résultats introversion-inhibition.*

En conclusion, au regard de ces résultats, nous constatons que soixante-douze pour cent (72%) de notre échantillon présente une attitude d'introversion-inhibition. A savoir, quarante pour cent (40%) sont plutôt introvertis et trente-deux pour cent (32%) sont tout à fait introvertis. Par contre, vingt pour cent (20%) des sujets sont pas ou peu introvertis alors que huit pour cent (8%) de nos sujets sont moyennement introvertis comme l'illustre parfaitement le graphique ci-dessus.

Compte tenu de ce constat, nous pouvons affirmer que nous avons travaillé avec des sujets introvertis. Il nous parait donc important de s'interroger sur l'apport de notre travail sur des sujets fortement inhibés et introverti.

Des recherches en neurophysiologie nous enseignent que le sujet présentant un comportement d'introversion-inhibition est un individu replié sur soi-même, plus réservé et moins bavard en groupe. Il est plus actif sur les activités telles que la lecture, les écrits, l'utilisation d'un ordinateur ou la balade. L'art, la musique, l'ingénierie, la sculpture et d'autres activités artistiques sont des professions hautement introverties. L'individu introverti prend plaisir en solitaire plutôt qu'en groupe, bien qu'il puisse apprécier les activités entre amis. Il préfère se concentrer sur une activité simple et observe les situations avant d'y participer. Les individus introvertis prennent le temps d'analyser avant d'agir. L'algorithmique est donc une activité qui cadre parfaitement avec des sujets introvertis. Notre échantillon étant constitué à soixante-douze pour cent de tels individus représente donc le choix de sujets le plus approprié pour notre expérimentation. Étant donné que les sujets de type introverti sont parfaitement adaptés à faire des activités en lien avec l'algorithmique, le faible niveau de performance générale de notre échantillon observé est dès lors difficilement explicable.

Quant au caractère d'inhibition, il n'est plus considéré comme une simple absence d'excitation, mais bien un processus actif de suppression d'une action excitatrice, comme le rappelle Christophe Boujon (2002). Il est important de signaler que les mécanismes de l'inhibition ont gagné récemment une place de choix dans la recherche, tant ses liens avec l'attention, la perception, l'intelligence, la mémoire et l'apprentissage sont forts et également au centre de tout activité de résolution de problème en algorithmique.

Pour Boujon (2002), l'inhibition intervient dans tout contexte ou situation nouvelle qui nécessite de l'attention. Comme l'attention, elle est plus lente à se mettre en place que les automatismes et la récupération des connaissances mémorisées. Elle permet avant tout de ne pas tenir compte, momentanément, d'éléments de la situation qui ne sont pas nécessaires pour réaliser une action, un comportement. Ce chercheur souligne que ce mécanisme est utile dans le sens où il permet de rendre disponibles et efficaces les processus d'analyse et de réponse des éléments pertinents. En bref, l'inhibition permet d'ignorer les informations parasites et de se centrer sur celles qui sont nécessaires pour agir, penser, etc. Elle agit comme un filtre. Cet aspect de l'inhibition nous interpelle particulièrement car présent dans toute situation de résolution de problème d'algorithmique. Le sujet est confronté à une double activité cognitive. Le filtre que constitue l'inhibition chez l'apprenant permettra à ce dernier de faire la bonne sélection des informations et de façon opportune. Le processus d'inhibition va permettre à l'apprenant de se concentrer sur ce qui est réellement nécessaire.

Finalement, l'apprenant doté d'un comportement d'introversion-inhibition est l'apprenant adapté pour les activités liées à la résolution de problème en algorithmique. En effet, cette activité à fort potentiel artistique requiert de la concentration, de la créativité, de l'attention, de l'intelligence, de la mémoire, l'utilisation de l'ordinateur, l'écriture, la lecture, etc.

## 3.4.2. Extraversion-Excitabilité-Instabilité

### - Tableau de synthèse des résultats

| | Pas | Peu | Moyennement | Plutôt | Tout à fait |
|---|---|---|---|---|---|
| S'agite constamment ou manipule des objets au cours de l'activité | 40 | 4 | 9 | 2 | 1 |
| Pose très fréquemment des questions après la lecture du sujet | 42 | 6 | 2 | 4 | 2 |
| Consulte d'autres sources d'information | 37 | 4 | 0 | 9 | 6 |
| N'accorde aucun soin à sa copie | 15 | 14 | 13 | 7 | 7 |
| Se met au travail rapidement | 28 | 15 | 3 | 8 | 2 |
| Participe très activement au travail demandé | 23 | 22 | 2 | 8 | 1 |
| Attitude de refus à l'invitation à se mettre au travail | 29 | 12 | 9 | 4 | 2 |
| Mène sa tâche rapidement, se contente d'une vue globale du sujet | 22 | 20 | 8 | 4 | 2 |
| Établi des similitudes avec d'autres tâches, d'autres sujets, d'autres exercices | 31 | 21 | 2 | 1 | 1 |
| Assume une responsabilité au début avec enthousiasme mais abandonne très vite | 20 | 28 | 2 | 1 | 5 |
| S'occupe beaucoup du travail des autres, soit en les aidant, soit en les critiquant | 24 | 18 | 9 | 3 | 2 |
| Parait adroit et vif | 19 | 29 | 5 | 2 | 1 |
| Est toujours partie prenante dans les groupes les plus actifs | 18 | 12 | 12 | 14 | 0 |
| Aborde le sujet avec excitation ou perd ses moyens à la découverte du sujet | 26 | 17 | 10 | 2 | 1 |
| Est très loin de percevoir la nécessité d'utiliser les règles syntaxiques dans la solution | 28 | 14 | 6 | 6 | 2 |
| **Total** | **402** | **236** | **92** | **75** | **35** |
| **pourcentage** | **48%** | **28%** | **11%** | **9%** | **4%** |

*Tableau 34 : tableau de synthèse des résultats extraversion-excitabilité-instabilité.*

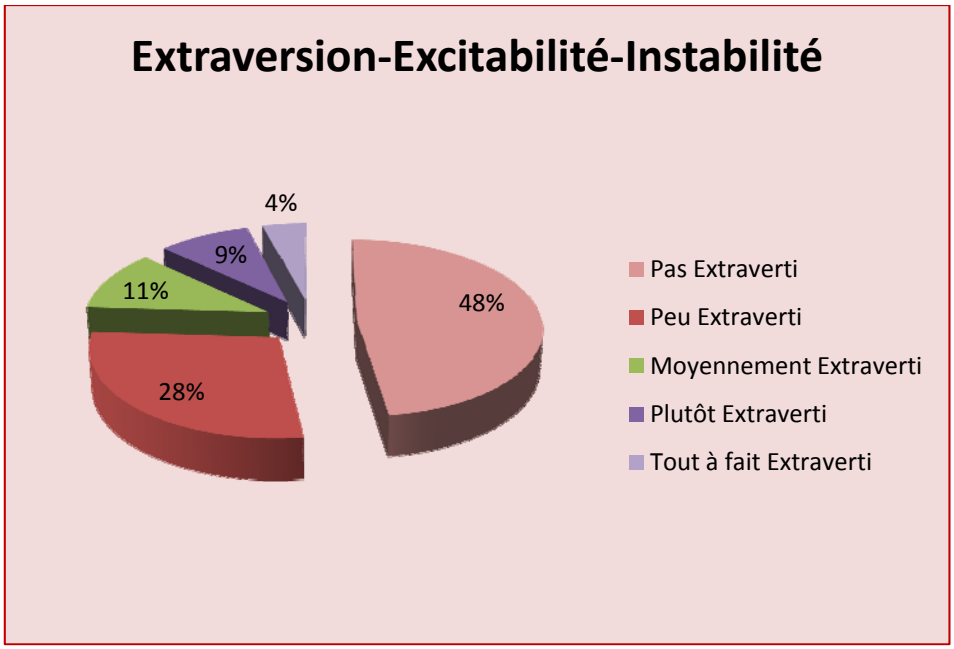



*Figure 41 : graphique de synthèse des résultats extraversion-excitabilité-instabilité.*

En définitive, concernant le caractère d'extraversion-excitabilité-instabilité, nous observons que quarante-huit pour cent (48%) des sujets sont pas extravertis, excités et instables. Vingt-huit pour cent (28%) sont peu extravertis, excités et instables, soit un total de soixante-seize pour cent (76%) de l'échantillon qui présente un comportement de non d'extraversion, non excitabilité et de stabilité, ainsi qu'en témoigne la figure 41.

Ce résultat confirme en quelque sorte le précédent car nous voyons que nos apprenants ne sont pas extravertis. Il n'y a que treize pour cent (13%) de notre échantillon qui est plutôt et tout à fait extravertis, instable et excité.

Finalement, les résultats de cette observation sont très cohérant. En effet, notre échantillon est à soixante-douze pour cent (72%) constitué d'individus introvertis, parfaitement aptes et recommandé pour des activités liées à l'algorithmique, ces individus sont aussi à soixante-seize pour cent (76%) non extravertis, non excités et stables. Ce qui en fait également des parfaits apprenants pour des apprentissages liés à l'ingénierie, l'art et la création, qui requiert la concentration, l'attention, etc., activités qui se trouve dans le registre de la résolution de problème en algorithmique.

En définitive notre échantillon était parfaitement adapté à notre cadre expérimental et son choix à été très efficient ce qui peut nous laisser dire que les résultats de notre expérimentation sont fiables ou du moins, la qualité ou les caractéristiques de notre échantillon ne peuvent pas constituer un élément de non fiabilité des résultats.

Après cette présentation de nos différentes expérimentations, des résultats aux quels nous sommes parvenus ainsi que des discussions pourtant sur nos résultats, nous allons consacrer cet ultime chapitre de notre recherche aux perspectives ouvertes par ce travail. Nous envisageons des perspectives dans le cadre du processus enseignement-

apprentissage ainsi que dans ce qui peut constituer une éventuelle ouverture des travaux de recherche initiés par cette thèse.

# 4. Perspectives et conclusion

## 4.1. Dans le processus enseignement-apprentissage

Au regard des résultats obtenus au cours de notre recherche, nous avons de plus en plus la conviction qu'une séance d'apprentissage de l'algorithmique et de la programmation ne saurait se dérouler sous forme d'un cours descendant, magistral, où les élèves resteraient passifs comme cela se fait actuellement à l'ENSET de Libreville et dans plusieurs autres établissement d'enseignement supérieur. Ainsi, il serait par exemple inefficace de demander à des élèves de reproduire durant un tiers de la séance un programme ou un algorithme que le professeur aurait expliqué au tableau durant les deux premiers tiers. Comme il est tout autant improductif pour un module de trente (30) heures de consacrer douze (12) heures au cours magistral, dix (10) heures aux travaux dirigés et huit (8) heures aux contrôles continus. Le caractère très abstrait de cette démarche pédagogique finit par démobiliser les apprenants car ils ne voient ni l'intérêt, ni le résultat concret de leur production.

Si chaque séance doit viser des objectifs de formation clairs et explicites, par exemple la déclaration des variables, il convient de réserver l'essentiel du temps à une activité autonome des élèves. Une séance peut commencer par quelques minutes où le professeur expose une situation-problème qui est une tâche concrète adaptée aux élèves pour qu'ils apprennent une notion précise : par exemple, il propose de reprendre un algorithme ou programme réalisé dans une séance précédente, en introduisant une nouvelle variable de même type que celle déjà là. Il montre comment déclarer une variable son rôle et comment elle tournera dans le programme ou l'algorithme. Puis il laisse les apprenants modifier leur programme ou algorithme, et peut proposer d'autres modifications (déclaration de variables de types différents) pour aller plus loin. La séance peut se terminer par la récapitulation des instructions permettant de déclarer ou d'utiliser les variables, éventuellement sous forme d'une fiche.

Le processus enseignement-apprentissage de l'algorithmique et de la programmation se prête particulièrement à la pédagogie de projet. L'objectif est de permettre à chaque élève de construire des connaissances et des compétences tout en développant son imagination et sa créativité dans le cadre d'un projet suivi sur quelques séances. Il nous semble important ici que l'enseignant comprenne que l'apprentissage de l'algorithmique et/ou de la programmation doit se traduire par la réalisation de productions soient individuelles soient collectives dans le cadre d'activités de création numérique (c'est un cours d'informatique), au cours de laquelle les apprenants développent leur autonomie, mais aussi le sens du travail collaboratif. Une approche possible serait qu'au cours d'une première séance, le professeur propose une activité, dont il a fixé clairement les objectifs de formation et les concepts nouveaux qui devront être installés. Une deuxième séance

peut permettre à chaque élève de développer son algorithme ou programme dans les directions qu'il aura lui-même choisies : la séance permet au professeur d'outiller l'élève selon ses besoins pour faire aboutir son projet. Il peut aussi aider l'élève à le redéfinir si la piste sur laquelle il s'est engagé est inadaptée. Une dernière séance peut permettre la finalisation des projets. Le professeur peut aussi proposer une mise en commun des concepts et des techniques utilisés par les uns et les autres. Ainsi dans le cadre d'un enseignement modulaire comme celui de l'ENSET de Libreville, l'enseignant pourra travailler sur trois (3) projets, chaque projet occupant trois (3) séances et chaque séance durant quatre (4) heures, soit trente-six (36) heures pour le module. Une note sera attribuée à chaque apprenant ou groupe d'apprenant à la fin de chaque projet, soit trois (3) notes à la fin du module.

Mettre en œuvre une pédagogie adaptée. Il s'agit d'amener chaque élève-professeur à la meilleure maîtrise possible de toutes ses attentes, dans le cadre d'un enseignement-apprentissage qui prend en compte ses acquis et ses marges de progression. Il s'agit d'accompagner chaque élève-professeur, en permettant aux meilleurs de construire des méthodes expertes dans la résolution d'un problème algorithmique ou de programmation, de développer des projets personnels ambitieux, d'assimiler des concepts d'algorithmique et de programmation riches. Il s'agit en même temps de conduire les élèves les plus en difficulté à une maîtrise suffisante des attentes. C'est ainsi qu'une séance peut comporter un problème permettant aux élèves qui auraient rapidement réalisé l'objectif commun de poursuivre en autonomie leur travail, pendant que le professeur se rend disponible auprès d'autres élèves. De même, comme on l'a dit pour la pédagogie de projet, le professeur guide l'élève dans la définition de son projet, en respectant ses choix, mais en les adaptant si besoin au degré de maîtrise effectivement atteint, que ce soit pour l'enrichir ou en réduire les objectifs.

## 4.2. Dans la poursuite de la recherche

Les résultats de notre étude ont montré que l'apport de notre artefact computationnel dans l'enseignement-apprentissage de l'algorithmique à l'ENSET de Libreville améliore les résultats des apprenants. Pour aller plus loin dans notre recherche, nous pouvons tester expérimentalement les démarches d'enseignement apprentissage proposées plus haut. Ces recherches auront pour buts de voire si l'utilisation d'un simulateur avec ces démarches facilite l'apprentissage de l'algorithmique. Plus concrètement, l'organisation d'une recherche autour de l'utilisation d'un simulateur d'algorithme dans le cadre d'une pédagogie adaptée pourrait être une piste. Une autre voie serait d'évaluer l'apport d'un simulateur d'algorithmes en pédagogie par projet ou en résolution de problèmes.

D'autres pistes de recherche seraient des études comparées non seulement des dispositifs enseignement-apprentissages selon les démarches pédagogiques énoncées précédemment mais aussi des différents outils de simulation d'algorithme avec des études approfondies de chacun d'entre-deux.

Afin d'ouvrir d'autres perspectives à ce travail de recherche, il nous semble important de signaler que notre analyse de la problématique de l'échec dans l'enseignement de l'algorithmique à l'ENSET de Libreville s'est fait principalement par rapport au processus enseignement-apprentissage. De ce point de vue, nous avons porté notre analyse sur la relation didactique et pédagogique ainsi que sur la relation d'apprentissage en nous appuyant sur les traces d'activités d'étudiants. Cependant, faire une analyse de l'échec ne peut se résumer uniquement à cela. Il nous semble donc plus judicieux pour compléter cette analyse d'observer les apprenants dans l'accomplissement d'une tâche prescrite, non plus par l'analyse des résultats de leur production, mais en tentant de comprendre leur démarche, leur stratégie mentale, leur sentiment, leur relation avec le savoir et l'artefact qu'il manipule, etc.

Toujours dans une perspective future de recherche, il est aussi envisageable de se lancer dans le développement d'un logiciel de simulation d'algorithmes répondant aux attentes spécifiques du cours d'algorithmique dans une institution de formation de formateurs de l'enseignement technique et professionnel. La plupart des logiciels de simulation actuellement disponibles sont destinés à l'enseignement secondaire et ont une approche beaucoup plus mathématique des algorithmes.

Une limite perceptible à notre dispositif est le fait que nous avons appliqué à tous nos sujets la même remédiation sans savoir au départ quel était leurs niveaux de connaissances respectifs. Une perspective de recherche possible serait l'élaboration d'un test-diagnostic afin d'adapter notre remédiation au niveau d'apprentissage des étudiants.

Il semble aussi opportun de faire une sorte d'analyse qui aurait pour but de comparer d'autres simulateurs afin de voire l'apport réel de chacun dans la facilitation de l'apprentissage. On pourrait se demander si la prise en main par les utilisateurs que sont l'enseignant et l'apprenant est aisée, l'usage nécessite t-il la présence d'un enseignant et le rôle de ce dernier, faut-il mettre en place une disposition particulière de l'espace classe (disposition des machine ainsi que le nombre d'apprenants par poste), etc. Tout ceci ayant pour but de trouver les conditions idoines à mettre en place afin que l'usage d'artefacts et la réactivation des savoirs implicite portés par la tache soient efficaces.

## 4.3. Conclusion

Le travail de recherche développé dans cette thèse s'articule autour de l'apport d'un logiciel de simulation d'algorithme dans le processus enseignement apprentissage de l'algorithmique chez les apprenants débutants de l'ENSET de Libreville au Gabon.

Il est important d'indiquer que l'apport d'un logiciel de simulation d'algorithme au service des pédagogies est un modèle efficient en matière d'intégration de cet outil dans les pratiques d'enseignement-apprentissage à l'ENSET. En effet, la nécessité de faire une place à ce dispositif en éducation ne semble plus un objet de débat en soi. La plupart des établissements d'enseignement supérieur technologique dans le cadre de leur curricula

se sont dotés de programmes en matière d'intégration de logiciel de simulation dans les cursus.

Nous avons indiqué que le processus enseignement-apprentissage est une activité chargée d'intentions telle que toute situation d'enseignement-apprentissage est nécessairement conçue dans l'esprit de l'enseignant. Ce dernier vise un certain nombre de buts et réalise des activités qui les préparent. Cette logique correspond aux actions indépendantes susceptibles de concevoir de bons outils à la fois intellectuels et matériels pour une amélioration de la construction de connaissances chez l'élève.

Nous avons également souligné que si l'on veut étudier et comprendre les activités de conception et d'utilisation réalisées respectivement par l'enseignant et l'élève, la nécessité d'observer attentivement les intentions de ces derniers aboutit au questionnement : quels buts poursuivent-ils consciemment ? Ici, il est concrètement question d'apporter des éléments de réponses au problème du fort taux d'échec dans l'enseignement-apprentissage de l'algorithmique à l'ENSET de Libreville. La solution que nous avons testée est d'introduire un artefact computationnel dans le processus.

Il est utile de rappeler que dans les situations d'enseignement-apprentissage classiques, dans lesquelles l'enseignant est le détenteur principal du savoir, il est possible d'envisager dans les moindres détails trois types d'intentionnalité. Il existe en conséquence autant de types de rôles dans la conception susceptible d'être en conflit les uns avec les autres. Il s'agit des rôles de l'enseignant-planificateur, de l'ingénieur-pédagogique et enfin celui de l'enseignant-en-classe. Ces trois niveaux d'intention existent dans toute situation d'enseignement-apprentissage classique. Ils sont rarement conflictuels puisque au regard de leur appartenance à une seule et même personne qui est l'enseignant. Devant ses élèves, ce dernier fait face au changement possible d'intentions entre sa préparation et son enseignement.

Il est à noter que des changements importants surviennent lorsque l'enseignement-apprentissage est médiatisé, non plus par l'enseignant, mais par des environnements divers. Cet aménagement fait naitre de nouveaux rôles, avec de types d'intention spécifique telle que le rôle du concepteur d'environnement ou le rôle d'enseignant-tuteur dont les intentions sont également variées : modérateur social, organisateur, facilitateur d'apprentissage et d'utilisation de l'environnement. Signalons d'une part que l'environnement incorpore très souvent des démarches pédagogiques aux intentions pouvant différer des trois premières notamment car elles dépendent de contraintes techniques de programmation informatique et d'autre part que le rôle d'enseignant-tuteur peut être joué par des étudiants avancés ou des enseignants.

Il est à préciser que l'objectif de notre travail n'est pas d'effectuer des modifications importantes sur l'apport d'un logiciel de simulation d'algorithme dans le processus enseignement-apprentissage de l'algorithmique. Par contre, le but que nous nous sommes fixés est de rendre plus actif nos apprenants, plus pratique et plus opérationnel l'enseignement-apprentissage de l'algorithmique, tout en conservant les aspects

classiques d'un enseignement. Cette démarche a été motivée par une analyse du passé scolaire de nos apprenants, leur demande et suggestion, leur désespoir devant les disciplines très théoriques. Nous avons souhaité qu'ils observent, simulent, touchent du doigt le résultat, ce que produit un algorithme. Sans intégrer les complexités de la programmation, ni aborder l'étude de notre outil, ce qui aurai crée d'autres problèmes méthodologiques et même didactiques. Il n'était pas question non plus de monter un dispositif numérique qui est une superposition de trois dispositifs, didactique, pédagogique et technique constituants trois instruments différents mais simultanément construit par l'apprenant (Marquet, 2010).

Il a été mis en exergue la motivation des apprenants par une remise en cause du sens irréel, mental, abstrait et théorique des algorithmes. Dans cette perspective, il est facile de percevoir la conformité d'un algorithme s'il produit le résultat souhaité. Tout en conservant un enseignement-apprentissage classique, l'utilisation d'un outil susceptible d'améliorer la construction des connaissances déclaratives et procédurales chez les apprenants a été notre dessein. Ce dispositif a finalement autorisé l'amélioration de leur performance. Dès lors, au cours d'une première expérimentation, le constat initial de difficultés des apprenants dans cette discipline en observant un taux d'échec inquiétant lors de notre test a été confirmé. Un ensemble de savoirs et savoir-faire sur lesquels les apprenants ont plus de difficultés a également été identifié. La seconde expérimentation permet de vérifier nos hypothèses. Soulignons que nous avons démontré que l'homogénéité de notre échantillon est un passage important, car les deux groupes en exercice doivent avoir le même niveau en algorithmique.

Au terme d'une analyse comportementale effectuée sur les sujets pendant les expérimentations, il a été démontré qu'ils ont un comportement d'introversion-inhibition. Cette réaction est une conduite idéale pour l'activité de résolution de problèmes en algorithmique. Leur choix est donc efficient. De ce fait, il ne peut pas constituer un élément de non fiabilité des résultats.

Ce travail présente deux ouvertures, l'une sur le plan pédagogique et l'autre dans la recherche. La mise en situation problème, la pédagogie de projet et la pratique d'une pédagogie adaptée constituent des perspectives intéressantes dans le processus enseignement-apprentissage. En effet, poursuivre cette recherche en mettant nos sujets non plus dans un dispositif classique d'enseignement-apprentissage, mais les plonger dans un véritable environnement de pédagogie active où ils vont construire leur propre connaissance en se servant du logiciel de simulation et voir ainsi l'apport du simulateur est une piste intéressante. Il est aussi à envisager dans le développement de ce travail de faire une étude comparée de deux ou de plusieurs logiciels de simulation d'algorithme. Cette étude mettra en exergue leur différence, leur similitude, leur complexité, leur avantage et inconvénient, etc. Cette étude permettra *in fine* de dégager des caractéristiques technologiques et pédagogiques importantes sur lesquelles il faut s'appuyer pour développer des applications de simulation d'algorithme. Le

développement d'applications en soit est aussi l'une des pistes de recherches à envisager.

Au regard de ces résultats, il est difficile d'affirmer de façon catégorique que le processus enseignement-apprentissage expérimenté est celui qui convient pour faciliter l'apprentissage de l'algorithmique. Le meilleur processus enseignement-apprentissage serait d'utiliser au mieux possible et de manière complémentaire les paramètres pertinents d'une situation d'apprentissage. Dans cette perspective, il est utile de commencer par déterminer les connaissances antérieures des apprenants, puis réaliser une analyse cognitive des tâches. C'est cette démarche qui a été envisagée. Concrètement, nous nous sommes attachés à étudier dans quelle mesure améliorer les résultats des apprenants. L'apport de notre application a dans une certaine mesure crée ce regain d'intérêt en produisant une légère amélioration des résultats des apprenants comme nous l'avons souhaité. Cependant, la réalité de l'expérimentation scientifique utilisée n'a pas fourni les écarts importants voulus. Cette différence est imputable à la faiblesse de notre échantillon et aux inconvénients associés au dispositif expérimental utilisé. Finalement, au regard de nos hypothèses de travail, nous avons pu démontrer qu'AlgoBox a facilité l'apprentissage de l'algorithmique en soulageant en partie l'apprenant de l'activité liée à l'apprentissage du formalisme par réduction de la charge cognitive intrinsèque. Il a été également prouvé qu'une réactivation des connaissances liées aux savoirs implicites portées par la tâche prescrite, et sollicitées dans l'élaboration de la solution, ont également facilité chez l'apprenant la compréhension de la situation-problème. Nous avons plus finement, en allant regarder les connaissances liées aux savoirs et savoir-faire implicites et explicites et dans les groupes concernés, démontré ces hypothèses.

# Bibliographie


Alessi, S., & Trollip, S. (2001). Multimédia pour l'apprentissage : méthode et développement, 3e édition. Pearson.

Amadieu, F., & Tricot, A. (2006). Utilisation d'un hypermédia et apprentissage : deux activités concurrentes ou complémentaires ? *Psychologie Française,* 51, 5-23.

André, B., Baron, G.-L., & Bruillard, E. (Éd.). (2004). Traitement de texte et production de documents : questions didactiques. Lyon : INRP, GÉDIAPS.

Astolfi, J. & Develay, M. (2002). Les concepts de la didactique des sciences. Dans *La didactique des sciences* (pp. 29-64). Paris: Presses Universitaires de France.

Baldy, R., Devichi, C., Aubert, F., Munier, V. Merle, H., Dusseau, J-M. & Fabre, J-F. (2005). Développement cognitif et apprentissage scolaire : l'exemple de l'acquisition du concept d'angle*. Revue française de pédagogie*, 152, 49-61.

Barma, S. (2007). Point de vue sur le nouveau programme science et technologie du secondaire au Québec : regards croisés sur les enjeux de part et d'autre de l'atlantique*. Revue didaskalia*, 30, 109-133.

Baron, G.L. & Bruillard, E. (2001). Une didactique de l'informatique ? *Revue française de pédagogie,* 135, 163-172.

Baron G.-L. (1989). L'informatique, discipline scolaire? (Le cas des lycées). Paris : PUF, 230 p. (Pédagogie d'aujourd'hui).

Baron, G.-L. (2016). Réflexions sur la didactique de l'informatique. *Adjectif.net* [En ligne] http://www.adjectif.net/spip/spip.php?article381

Baron G.-L., Bruillard, E. (1996). L'informatique et ses usagers dans l'éducation. Paris : PUF, 312 p. (L'éducateur).

Baron G.-L., Bruillard, E., Levy, J.-F. (dir.) (2000). Les Technologies dans la classe : De l'innovation à l'intégration. Paris : INRP : EPI, 199 p. : ill.,bibliogr.

Baron, G.-L., Baudé, J., & Cornu, P. (Éd.). (1989). Colloque francophone sur la didactique de l'informatique. Paris : EPI. Consulté à l'adresse http://edutice.archives-ouvertes.fr/edutice-00374950

Baron, G.-L., Bruillard, E., & Drot-Delange, B. (Éd.). (2015). Informatique en éducation : perspectives curriculaires et didactiques. Clermont-Ferrand, France : Presses universitaires Blaise-Pascal.

Baron, G.-L., Bruillard, E., & Komis, V. (Éd.). (2011). Didapro 4 - Dida&STIC. Sciences et technologies de l'information et de la communication (STIC) en milieu éducatif Analyse de pratiques et enjeux didactiques. Athènes : new technologies editions.

Bastien, C. (1997). Les connaissances de l'enfant à l'adulte. Paris: Armand Colin.

Bastien, C. & Bastien-Toniazzo, M. (2005). Du cheminement aux cheminements…. *Revue française de pédagogie*, 152, 21-28.

Bautier, É. & Rochex, J.-Y. (2007). "Apprendre, des malentendus qui font la différence". In J. Deauvieau & J.-P. Terrail (Éd.). *Les sociologues, l'école et la transmission des savoirs*. Paris : La Dispute, pp. 227-243.

Bekale Nze, J.S., Ginestié, J., & al (2014). Educational in East and Central Africa. London: Bloomsburry Academic.

Beliste, C. (2010). Les technologies : quels usages, pour quels effets. In B. Charlier & F. Henri (Eds). Apprendre avec les technologies, (pp. 35-45). Paris: puf.

Benedetto, P. (2013). Psychologie cognitive : concepts fondamentaux. Paris: Studyrama.

Bénézra, A., Jean, F., Rothan, B. (1989). Six modes d'utilisation de l'ordinateur pour l'enseignement des disciplines. Bulletin de l'EPI (Enseignement Public et Informatique), (55). 100-108.

Biagioli, N. (2014). Didactique(s) : un singulier-pluriel. Réaction aux points de vue développés. *Revue éducation & didactique, vol.8. 140, 45-51*.

Bibeau, R. (1996). École informatisée clés en main. Projet franco-québécois de recherche-action. *Revue de l'EPI (Enseignement Public et Informatique)*, (82), 137-147.

Bibeau, R. (2007). Les technologies de l'information et de la communication peuvent contribuer à améliorer les résultats scolaires des élèves. *Revue de l'EPI*, (94).

Brandt-Pomares, P. & Boilevin, J. M. (2008). Didactique des sciences physique, didactique de la technologie et usage des TICE. Journal of e-Learning and knowledge society, 4(2), 245-254.

Brousseau, G. (1998). Théorie des situations didactiques, la pensé sauvage. Grenoble.

Brown, M.H. (1988). Exploring algorithms using Balsa II, IEEE Computer, vol 21, n°5, pp14-36.

Bruillard, E. (2000). « Qu'importe qu'ils comprennent puisqu'ils savent s'en servir ! » Les dossiers de l'ingénierie éducative, n° 31, Paris, CNDP, p. 2-3.

Bruillard, E. & Baron, G.-L. (2006). Usages en milieu scolaire: caractérisation, observation et évaluation (pp. 269-284). In M. Grandbastien, & J.-M. Labat, (Eds), *Environnements informatiques pour l'apprentissage humain*. Paris: Lavoisier.

Bruner, J. S. (1996). L'éducation, entrée dans la culture. Paris: Retz.

Buty, C. & Badreddine, Z. (2009). Quelques effets didactico-discursifs de l'utilisation des schémas : cas d'un enseignement d'électricité*. Revue Aster*, 48, 87-110.

Bybee, R. W, Powell, J. C. et Ellis, J. D. (1991). Integrating the history and nature of science and technology in science and social studies curriculum. Science Education, 75(1), 143-155.

Bygate, M., Skehan, P. & Swain, M. (2001). "*Researching pedagogic tasks second language learning, teaching and testing"*. Cambridge: Cambridge University Press.

Cartwright, N. (1983): la façon dont se déroulent les lois de la physique. Oxford, New York: Clarendon Press.

Cartwright, N. (1999): The Dappled World. Une étude des frontières de la science. Cambridge: Cambridge University Press.

Cauzimille-Marmèche, E. (1991). Apprendre à utiliser ses connaissances pour la résolution de problèmes : analogie et transfert, *Bulletin de psychologie,* 44, 156-164.

Cauzimille-Marmèche, E. (1996). Effet du rôle assigné à l'expert dans la résolution en dyade asymétrique d'une tâche de combinatoire. *Archives de psychologie,* 64, 109-131.

Charlier, B. (2010). Les TIC ont-elles transformé l'enseignement et la formation. In B. Charlier & F. Henri (Eds). *A*pprendre avec les technologies, (pp. 146-156). Paris: puf.

Chevallard, Y. (1985). La transposition didactique : savoir savant, savoir enseigné. La pensé sauvage. Grenoble.

Chevallard, Y. (2014). Des disciplines scolaires à la didactique comme science anthropologique sur des obstacles épistémologique, psychologique et institutionnel. *Revue éducation & didactique, vol.8. 140, 35-43*.

Clement, J., Brown, D. E. et Zietsman, A. (1989). Not all preconceptions are misconceptions: Finding "anchoring conceptions" for grounding instruction on students' intuitions. *International Journal of Science Education, 11(5),* 554-565.

Crookes, G. (1986). "Task classification: A cross-disciplinary review". *Technical Report, n°4*, The Center for Second Language Classroom Research, Social Science Research Institute,University of Hawai: Manoa.

Crozat, S. (2002). Éléments pour la conception industrialisée des supports pédagogiques numériques (Doctoral dissertation, Université de Technologie de Compiègne).

Da Costa, J. (2014). BPMN 2.0 pour la modélisation et l'implémentation de dispositifs pédagogiques orientés processus (Doctoral dissertation, University of Geneva).

De Jong, & Al. (1999). The integration of computer simulation and learning support: An example from the physics domain of collision. Journal of research in science teaching.

Depover, C. (1987). L'ordinateur média d'enseignement, cadre conceptuel, problématique et recherches. Bruxelles: Edition De Boeck.

Depover, C., Giardina, M. & Marton, P. (1998). Les environnements d'apprentissage multimédia : analyse et conception. Paris: L'harmattan.

Denis, B. (1993). Les animateurs développent-ils une pédagogie constructiviste en robotique pédagogique? In Regards sur la robotique pédagogique: actes du 4e colloque international sur la robotique pédagogique. INRP.

Dessus, P. (2010). Des théories de l'apprentissage pour concevoir des environnements d'apprentissage informatisés. In B. Charlier & F. Henri (Eds). *Apprendre avec les technologies*, (pp. 95-107). Paris: puf.

D'Hainaut, L. (2000). Des fins aux objectifs de l'éducation. Bruxelles: Edition Labor, Collection Éducation.

Doolittle, P. E., (1999). Constructivism and online education. Virginia: Polytechnic Institute & State University.

Driver, R. (1989). Students' conceptions and the learning of science. International Journal of Science Education 1(5), 481-490.

Duchâteau, C. (1994). Faut-il enseigner l'informatique à ses utilisateurs. In Québec, Communication au quatrième colloque francophone sur la didactique de l'informatique. Consulté à l'adresse https://pure.fundp.ac.be/ws/files/988521/54322.pdf

Duchâteau, C. (2003). Peut-on enseigner les « outils » logiciels ? Un dispositif pour une auto-formation au traitement de texte, balisée et assistée, partiellement à distance : analyse d'une expérience. In Premières journées francophones de didactique des progiciels Didapro. Consulté à l'adresse http://edutice.archives-ouvertes.fr/edutice-00145560/en/

Duplàa, E., & Talaat, N. (2012). Connectivisme et formation en ligne. *Distances et savoirs*, *9*(4), 541-564

Eysenck, H.J. (1960). The structure of human personality. (Second Edition). London: Methuen

Eysenck, H.J. (1967). The biolooica1 basis of personality. Springfield: Thomas.

Finance, J.-P. (2000). – Entre rétrospective et prospective. *In* René Jacquart (dir.), Informatiques. Enjeux, tendances et évolutions. Paris : Hermès, p. 11-17.

Fourez, G. (1992). La construction des sciences. Les logiques des inventions scientifiques. Bruxelles : De Boeck.

Fourez, G. et Lambert, D. (1989). À chacun ses modèles théoriques. In G. Fourez (dir.), Enseigner les sciences en l'an 2 000. Symposium international, enseignement des mathématiques et des sciences, technologies, éthiques et sociétés. Presses universitaires de Namur.

Frenay, M. & Galand, B. (2007). Les dispositifs d'apprentissages par problèmes dans l'enseignement supérieur. In V. Dupriez & G. Chapelle (Eds). Enseigner, (pp. 117-128). Paris: puf.

Freinet, C. (1992). Œuvres pédagogiques, tome 2. Paris : S*euil.*

Furio Mas, C-J., Iturbe Barrenetxea, J. & Reyes Martin, J-V. (1994). La "résolution de problèmes comme recherche" : une contribution au paradigme constructiviste de l'apprentissage des sciences*. Revue Aster*, 19, 87-102.

Gagné, E. D. (1985). The cognitive psychologie of school learning. Boston: Little, Brown and Company.

Gauthier, C., Bissonnette, S., & Richard, M. (2007). L'enseignement explicite. In V. Dupriez & G. Chapelle (Eds). *Enseigner*, (pp. 106-116). Paris : puf.

Gilbert, JK., Boulter, CJ., & Elmer, R. (2000). Modèles de positionnement dans l'enseignement des sciences et dans la conception et l'éducation technologique. In JK. Gilbert et CJ. Boulter (Eds.), *Développer des modèles dans l'enseignement des sciences* (pp. 3-17). Dordrecht, Pays-Bas: Kluwer Academic Publishers.

Ginestié, J. (1996). Computer based control in Technology Education: Some questions about introducing and teaching. In A. Tamir (dir.). Report of the Jerusalem International Sciences and Technology Education Conference. Section 3, 21-29, Israel, Jerusalem: UNESCO.

Ginestié, J. & Andreucci, C. (1997). Approach of assessment and teaching meaningful in Technology education in France: tries and mistakes. 8th PATT Conference, 10-14 April, Breukelen, PATT Foundation

Ginestié, J., (2008). From task to activity: a re-distribution of role between teacher and pupils. In J. Ginestié (Ed.), The cultural transmission of artefacts, skills and knowledge: Eleven studies in technology education (pp. 225-257). Rotterdam: Sense Publishers.

Giordan A. (1990). Apprendre, comprendre, s'approprier l'environnement. Cahiers pédagogiques, 312, 35-37.

Giordan, A. (1991). La modélisation dans l'enseignement et la vulgarisation des sciences. Impact: science et société, 41(4), 337-355.

Giordan, A. et De Vecchi, G. (1987). Les origines du savoir. Neuchâtel et Paris: Delachaux et Niestlé

Glaser, R. (1987). Further notes toward a psychology of instruction. In R. Glaser (Ed.), *Advances in instructional psychology* (Vol. 3, pp 1–39). Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Good, T. et Brophy, J. (1995). Educational Psychology: A Realistic Approach, 4e éd., New York : Longman.

Gray, J.A., (1967). Strength of the nervous system, introversion extraversion, conditionability and arousal. Behavior research & therapv, (pp.151-169).

Hamon, C. (2009). Graphismes techniques : tâches, nature et causes des difficultés des apprenants. *Revue Aster*, 48, 39-62.

Hannoun Kummer, P. (2009). Résolutions graphiques et mécanique : caractère en contexte. *Revue Aster*, 48, 133-160.

Hernert, P. (2002). Les algorithmes, Paris, Presses universitaires de France. *Que sais-je ?.*

Hoc, J.-M. (1984). « Les activités de résolution de problèmes dans la programmation », Psychologie française. Paris, Armand Collin, 29, 3/4, 267-271.

Hodson, D. (1988). Experiments in science and science teaching. Educational Philosophy and Theory, 20, 53–66.

Hodson, D. (1992). In search of a meaningful relationship: An exploration of some issues relating to integration in science and science education. International Journal of Science Education 14(5), 541-562.

Jacquart, R. (dir.) (2000). – Informatiques. Enjeux, tendances, évolution. Numéro spécial Technique et science informatiques. Paris : Hermès, 467 p.

Johsua, S. & Dupin, J-J. (1993). Introduction à la didactique des sciences et des mathématiques. Paris: puf.

Kalyuga, S., Chandler, P. & Sweller, J. (1998). Levels of expertise and instructional design. Human Factors, 40, 1-17.

Kitajima, J-P. (2004). Modèles quantitatifs d'algorithmes parallèles (Thèse de doctorat), Institut National Polytechnique de Grenoble, Grenoble.

Knuth, D.E. (1981). The art of computer programming, volume 2: Seminumerical algoriths. Massachusetts: Addisso-Wesley, 2nd edition.

Kokou, A. (2010). Médias et technologies dans les pays en développement. In B. Charlier & F. Henri (Eds). *Apprendre avec les technologies*, (pp. 81-94). Paris : puf.

Landa, L.N. (1962). L'enseignement aux élèves des écoles des méthodes de pensée rationnelle et le problème des algorithmes. Vop. Psych.

Landa, L.N. (1966). Recherche sur l'application de logique mathématique et de la théorie d'informatique à quelque problème d'enseignement. Vop. Psych. 2.

Landa, L.N. (1966). Algorithmes dans l'enseignement. Moscow, éditions du Progrès.

Larochelle, M. et Desautels, J. (1992). Autour de l'idée de science. Sainte-Foy et Belgique: Les Presses de l'Université Laval et De Boeck-Wesmàel.

Laveaut, D., Grégoire, J. (2014). Introduction aux théories des tests en psychologie et en sciences de l'éducation. Bruxelles: De Boeck supérieur.

Lawson, A. E. (1985). A review of research on formal reasoning and science teaching. Journal of Research in Science Teaching, 22(7), 569-61

Legendre, M.-F. (1994). Problématique de l'apprentissage et de l'enseignement des sciences au secondaire : un état de la question. *Revue des sciences de l'éducation 204*, 657– 677.

Levy, J.-F. (dir.) (1995). – Pour une utilisation raisonnée de l'ordinateur dans l'enseignement secondaire; analyses de pratiques et propositions pour un meilleur usage des instruments informatiques. Paris : EPI ; INRP, 178 p. (Didactiques des disciplines).

Linder, C. J. (1992). Is teacher-reflected epistemology a source of conceptual difficulty in physics? International Journal of Science Education, 14, 11-121.

Lucas, M. (1994). – Enseignement d'un ensemble de notions et savoir-faire informatiques à l'école primaire, au collège et au lycée. *In* Informatique, formation des enseignants : quelles interactions ? Paris : INRP, p. 75-90

Maddux, C.D., Johnson, D.L., & Willis, J.W. (2001). Educational computing: Learning with tomorrow's technologies. Boston: Allyn and Bacon.

Malgouyres, R., Zrour, R. & Fescher, F. (2011). Initiation à l'algorithmique et à la programmation en C : cours avec 129 exercices corrigés. Paris : Dunod.

Malone, TW (1981). Vers une théorie de l'instruction à motivation intrinsèque. *Cognitive Science*, *5* (4), 333-369.

Marquet, P. (2010). Apprendre en construisant ses propres instruments. In B. Charlier & F. Henri (Eds). *Apprendre avec les technologies*, (pp. 119-129). Paris : puf.

Martinand, J-L. (1986). Connaitre et transformer la matière : des objets pour l'initiation aux sciences et techniques. Berne : Peter Lang.

Martinand, J-L. (1994). La didactique des sciences et de la technologie et de la formation des enseignants*. Revue Aster*, 19, 61-75.

Martinand J.-L. (1994). In Dictionnaire encyclopédique de l'éducation et de la formation, Paris, Nathan, pp.255-256.

Maryvonne, M. (2007). Activité humaine et conceptualisation. Questions à Gérard Vergnaud, Presses Universitaire du Mirail (374 p.).

Mayer, R.E. (2001). Apprentissage multimedia, New York: Cambridge University Press.

Mayer, R.E. (2005). Cambridge Handbook of Multimedia Learning. New-York, USA: Cambridge University Press.

Mayer, R.E. (2008). Learning and Instruction. Upper Saddle River (New-York), USA: Prentice Hall.

Mayer, R.E. (2009). Multimedia Learning, second edition, Cambridge: Cambridge University Press

Meirieu, Ph. (1987). "*Apprendre... oui, mais comment*". Paris, ESF.

Meirieu, Ph., Develay, M., Durand, C, et Mariani, Y. (dir.) (1996). Le concept de transfert de connaissance en formation initiale et continue, Lyon, CRDP.

Meirieu, Ph. (1995*). La Pédagogie entre le dire et le faire*. Paris, ESF.

Meirieu, Ph. (2004). *Faire l'école, faire la classe*. Paris, ESF.

Nicolas, S. & Perruchet, P. (1998). "L'apprentissage implicite : un débat théorique". *Psychologie française* n°43/1, pp. 13-25.

Nijimbere, C. (2015). L'enseignement de savoirs informatiques pour débutants, du second cycle de la scolarité secondaire scientifique à l'université en France. Consulté à l'adresse http://www.adjectif.net/spip/spip.php?article355&lang=fr

Orange, C. (1991). Didactique de l'informatique et pratiques sociales de référence. EPI, bulletin n° 60, décembre 1990, p. 151-161.

Orange, C. (2014). Regard complémentaire. Unité et diversité du didactique. *Revue éducation & didactique, vol.8. 140, 85-90*.

Paas, F., & Van Merriënboer, J. (1994). Instructional control of cognitive load. In the training of complex cognitive tasks. Educational Psychology Review, 6, 351-371.

Padilla, M. J. (1991). Science activities, process skills, and thinking. In S. M. Glynn, R. H. Yeany et B. K. Britton (éd.), The psychology of learning science. Hillsdale, NJ: Lawrence Erlbaum Associates.

Paindorge, M. (2007). La progressivité des notions dans les programmes de l'éducation technologique*. Revue didaskalia*, 30, 89-108.

Papert, S. (1981). – Le jaillissement de l'esprit. Paris : Flammarion.

Parrochia, D. (2000). La raison systématique. Vrin.

Pastré, P., Parage, P., Richard, JF., Sander, E., Labat, JM., Futtersack, M. (2009). La résolution de problèmes professionnels sur simulateur. *Activités, 6*, 3-28.

Peraya, D. (2010). Médias et technologies dans l'apprentissage : apports et conflits. In B. Charlier & F. Henri (Eds). *Apprendre avec les technologies*, (pp. 24-34). Paris: puf.

Prabhu, N. S. (1987). "*Second language pedagogy*". Oxford: Oxford University Press.

Prensky, M. (2001). Digital Natives, Digital Immigrants Part 1, On the Horizon, Vol. 9 Issue: 5, pp.1-6, https:// doi.org/10.1108/10748120110424816.

Piaget, J. (1972). Où va l'éducation? (2e éd.). Paris: Gonthiers, Denoël (lre éd., 1948).

Piaget, J. (1975). L'équilibration des structures cognitives, P.U.F., Paris.

Pochon, L.-O., Bruillard, E., & Maréchal, A. (2006). Apprendre (avec) les progiciels. Entre apprentissage scolaire et pratiques professionnelles. Neuchâtel / Lyon: IRDP ; INRP.

Ponsonnet, L. (2011). Initiation à l'algorithmique et à la programmation pour le Lycée. Paris : ellipses.

Reboul, O. (1980). Qu'est-ce qu'apprendre ?. Paris : PUF.

Reuter, Y. (2010). Dictionnaire des concepts fondamentaux des didactiques. Bruxelles : *De Boeck Supérieur.*

Reuter, Y. (2014). Didactique et discipline : une relation structurale. *Revue éducation & didactique, vol.8. 140, 53-64*.

Richard, J-F. (1982). « Planification et organisation des actions dans la résolution du problème de la tour de Hanoï par des enfants de 7 ans », l'année psychologique, 82, 307-336.

Richard, J-F. (1994). « Raisonnement pour l'action : résolution de problème ». in Ghiglione R., Richard JF. (éd). Cours de psychologie 3 : Champs et théories. Paris, Dunod.

Richard, J-F. (2004). Les activités mentales : de l'interprétation de l'information à l'action. Paris : Armand Colin.

Richard, J-F. (2005). Les activités mentales en résolution de problèmes : comprendre, raisonner, trouver une solution. Paris: Armand Colin.

Richard, J-F. & al. (1993). Problem solving restructuration: Elimination of implicit constraints. Cognitive Sciences, 17, 497-529.

Richards, J., Platt, J. & Weber, H. (1985). "*Longman dictionary of applied linguistics".* Harlow, Essex: Longman.

Rogalski, J. (1990). – Didactique de l'informatique et acquisition de la programmation. Recherches en didactique des mathématiques, vol. 9, n° 3, p. 407-425.

Rogalski, J., & Samurcay, R. (1986). – Les Problèmes cognitifs rencontrés par des élèves de l'enseignement secondaire dans l'apprentissage de l'informatique. Journal Européen de Psychologie de l'Éducation, vol. 1, n° 2, p. 97-110.

Rouet, J-F. (2005). La conception des ressources multimédias pour l'apprentissage : apports des recherches en psychologie du langage. Revue française de pédagogie, 152, 21-28.

Ruggieri, C, Tarsitani, C. et Vicentini, M. (1993). The images of science of teachers in Latin countries. International Journal of Science Education, 15(4), 383-393

Sander, E. (1997). Analogie et catégorisation, Thèse de Doctorat de psychologie. Université de Paris VIII.

Saujat, F. (2007). Enseigner : un travail. In V. Dupriez & G. Chapelle (Eds). *Enseigner*, (pp. 179-188). Paris : puf.

Schneuwly, B. (2014). Didactique, construction d'un champ disciplinaire. *Revue éducation & didactique, vol.8. 140, 53-64*.

Schnotz, W. (2008). Why multimedia learning is not always helpful. In J.F. Rouet, R. Lowe & W. Schnotz (Eds.),

Schulman, L. S. (1986). Those who understand: Knowledge growth in teaching. Educational *Researcher, 15(2),* 4-14.

Siemens, G. (2005). Connectivism : A learning theory for the digital age. International journal of instructional technology and distance learning, *2*(1), 3-10.

Sweller, J. (1988). Cognitive load during problem solving: Effects on learning. Cognitive Science*, 12*, 257–285.

Sweller, J. (2010). Element interactivity and intrinsic, extraneous, and germane cognitive load. Educational Psychology Review*, 22*, 123-138.

Sweller, J. & Chandler, P. (1994). Why some material is difficult to learn. *Cognition and Instruction, 12(3), 85–233.*

Szczygielsi, C. (2009). Lecture et compréhension dans différents systèmes sémiotiques en électricité : raisonner sur des schémas électrocinétiques ou électrotechniques et des montages électriques. Revue Aster, 48, 161-186.

Tardif, J. (1992), Pour un enseignement stratégique, l'apport de la psychologie cognitive. Montréal, Logiques.

Tardif, J. (1995), Savoirs et savoir-faire : une dynamique pédagogiquement ignorée, in Bentolila, A. (dir.) *Savoirs et savoir-faire*, Paris, Nathan, p. 89-104.

Tricot, A., Plegat-Soutjis, F., Campus, J-F., Amiel, A., Lutz, G., & Morcillo, A. (2003). Utilité, utilisabilité, acceptabilité : interpréter les relations entre trois dimensions de l'évaluation des EIAH. In C. Desmoulins, P. Marquet & D. Bouhineau (Eds). *Environnement Informatique pour l'Apprentissage Humain*, (pp. 391-402). Paris : ATIEF/INRP.

Tricot, A. (2003). Apprentissage et recherche d'information dans des documents électroniques. Mémoire pour l'Habilitation à Diriger des Recherches, Université de Toulouse Le Mirail (128 p.).

Van Lehn. (1990). Mind bugs: origins of procedural misconception. Cambridge. Mass: MIT Press.

Vergnaud, G. (1968). Pour un model algorithmique du sujet. Doc. Ronéotype.

Vergnaud, G. (1991). Langage et pensée dans l'apprentissage des mathématiques. Revue Française de Pédagogie, 96, 79-86.

Vergnaud, G. (1991). La théorie des champs conceptuels. Recherche en didactique des mathématiques, 10, 133-170.

Vergnaud, G. (1994), Apprentissages et didactiques, où en est-on *?*, Paris, Hachette.

Vergnaud, G. (2001). Psychologie du développement cognitif et évaluation des compétences. In L'activité évaluative réinterrogée. Regards scolaires et socioprofessionnels. pp. 43-51.

Vergnaud, G. (2002). L'explication est-elle autre chose que la conception ?. In F. Leutenegger & M. Saada-Robert (Eds). *Expliquer et comprendre en sciences de l'éducation*. Bruxelles : de Boeck.

Viennot, L. (1979). Le raisonnement spontané en dynamique élémentaire. Paris: Hermann

Villiot-Leclercq E. (2007). La méthode des Pléiades : un formalisme pour favoriser la transférabilité et l'instrumentation des scénarios pédagogiques. *Revue STICEF (Sciences et Technologies de l'Information et de la Communication pour l'Éducation et la Formation)*, Vol.14.

Vygotsky, L. (1978). - Mind in society, Harvard University Press, Cambridge, Massachussetts.

Vygotsky, L. (1980). Mind in society: The development of higher psychological processes. Cambridge, MA: Harvard University Press.

Watson, J. (1972). Le béhaviorisme. Paris. Editions Cepi.

Wenger, E. (1987). *Systèmes d'intelligence artificielle et de tutorat: approches computationnelles et cognitives de la communication des connaissances. S*an Francisco: Morgan Kaufmann.

Wilensky U. (Eds). (2003). Special Issue on Agent-Based Modeling. International Journal of Computers for Mathematical Learning, 8/1.

# Index des figures

# Index des tableaux

# Sigles et abbreviations

| | |
|---|---|
| **AAO** | *Apprentissage Assisté par ordinateur* |
| **AFCET** | *Association française pour la cybernétique économique et technique* |
| **AFDI** | *l'Association Francophone de Didactique de l'Informatique* |
| **B2i** | *Brevet informatique et internet* |
| **BT** | *Brevet de Technicien* |
| **C2i** | *Certificat informatique et internet* |
| **CNRS** | *Centre national de la recherche scientifique* |
| **CPU** | *Central Processing Unit* |
| **DDL** | *Degré de liberté* |
| **Didapro** | *Didactique des Progiciels* |
| **EAO** | *Enseignement Assisté par Ordinateur* |
| **ENSET** | *École Normale Supérieure de l'Enseignement Technique* |
| **EPI** | *Enseignement Public et l'Informatique* |
| **ERASMUS** | *European Action Scheme for the Mobility of University Students* |
| **F** | *Sexe féminin* |
| **F1** | *Génie mécanique, option productique mécanique* |
| **F1D** | *Génie Mécanique, option Bois et Matériaux Associés* |
| **F2** | *Génie électrique, option électronique* |
| **F3** | *Génie électrique, option électrotechnique* |
| **F4** | *Génie civil, option construction bâtiment* |
| **GE** | *Groupe Expérimental* |
| **GT** | *Groupe Témoin* |
| **GUI** | *Graphic User Interface* |
| **$H_0$** | *Hypothèse nulle* |
| **$H_a$** | *Hypothèse alternative* |
| **HTML** | *HyperText Markup Language* |
| **http** | *HyperText Transfer Protocol* |
| **https** | *HyperText Transfer Protocol Secure* |
| **IHM** | *interfaces homme-machine* |
| **LARP** | *Logiciel d'Algorithmes et de Résolution de Problèmes* |
| **LDA** | *Langage de description d'Algorithmes* |
| **LOGO** | *Le nom « Logo » vient du grec « Logos » qui signifie : parole, discours, intelligence* |
| **LTNOB** | *Lycée Technique National Omar Bongo* |
| **M** | *Sexe masculin* |
| **MI** | *Maintenance Industrielle* |
| **MIT** | *Massachusetts Institute of Technology* |
| **MoyGE** | *Moyenne groupe expérimental* |

| | |
|---|---|
| **MoyGT** | *Moyenne groupe témoin* |
| **MoyPost test** | *Moyenne au post test* |
| **MoyPré-test** | *Moyenne au pré-test* |
| **MVA** | *Maintenance de véhicule automobile* |
| **PDF** | *Portable Document Format* |
| **PGCD** | *Plus Grand Commun Diviseur* |
| **R** | *Réponse* |
| **RAM** | *Random Access memory* |
| **S** | *Savoir* |
| **SE** | *Savoir Explicite* |
| **SF** | *Savoir-faire* |
| **SI** | *Savoir Implicite* |
| **SFE** | *Savoir-faire Explicite* |
| **SFI** | *Savoir-faire Implicite* |
| **SM** | *Structures Métalliques* |
| **sqr** | *Square (la fonction carrée)* |
| **sqrt** | *Square-root (la fonction racine carrée)* |
| **STI** | *Sciences et techniques industrielles* |
| **STT** | *Sciences et techniques du tertiaire* |
| **TIC** | *Technologies de l'Information et de la Communication* |
| **TICE** | *Technologies de l'Information et de la Communication pour l'enseignement* |
| **UOB** | *Université Omar Bongo* |
| **URL** | *Uniform Resource Locator* |
| **VAR(GE)** | *Variance groupe expérimental* |
| **VAR(GT)** | *Variance groupe témoin* |
| **WWW** | *World Wide Web* |

# Annexes

| Annexes | Désignations |
|---|---|
| Annexe 1 | Test de la première expérimentation et du pré-test. |
| Annexe 2 | Exemple de production d'un apprenant au pré-test. |
| Annexe 3 | Test proposé au post test. |
| Annexe 4 | Exemple de production d'un apprenant au post-test. |
| Annexe 5 | Tableau de codification des productions des apprenants. |
| Annexe 6 | Liste des savoirs explicites et implicites par exercice. |
| Annexe 7 | - Grille d'observation Introversion/inhibition.<br>- Résultats obtenus par tous nos sujets. |
| Annexe 8 | Grille d'observation Introversion/inhibition (complétée). |
| Annexe 9 | - Grille d'observation Excitabilité/instabilité/extraversion.<br>- Résultats obtenus par tous nos sujets. |
| Annexe 10 | Grille d'observation Excitabilité/instabilité/extraversion (complétée). |
| Annexe 11 | Cursus de formation à l'ENSET de Libreville. |
| Annexe 12 | Énoncés et propositions de solutions |

**ANNEXE 1 :** Test de la première expérimentation et du pré-test

# ALGORITHMIQUE
# LICENCE 1 – STI
## 2 heures (documents autorisés)

**Exercice N°1**

Écrire un algorithme qui lit un nombre entier et vous affiche après vérification si ce nombre est pair ou impair.

**Réponse**

**Exercice N°2**

Soit une équation du second degré de la forme $Ax^2 + Bx + C = 0$.
Écrire un algorithme qui lit A, B et C au clavier et affiche les éventuelles solutions après s'être assuré que l'équation est bien du second degré.

**Réponse**

**Exercice N°3**

Écrire un algorithme qui calcule puis affiche tous les carrés des entiers inférieurs ou équivalents à 10.

**Réponse**

**Exercice N°4**

Écrire un algorithme qui calcule le Plus Grand Commun Diviseur (PGCD) de deux entiers naturels par la méthode des différences successives.

**Réponse**

**ANNEXE 2 :** Exemple de production d'un apprenant au pré-test

S38 5

# ALGORITHMIQUE
## LICENCE 1 – STI
2 heures (documents autorisés)

**Exercice N°1**
Écrire un algorithme qui lit un nombre entier et vous affiche après vérification si ce nombre est pair ou impair.

**Réponse**

Programme AFFICHAGE

Variable N, P, I : réels

début
lire (N)
Si N ← 2
écrire N ← P
sinon
écrire N ← I
fSi

**Exercice N°2**
Soit une équation du second degré de la forme $Ax^2 + Bx + C = 0$.
Écrire un algorithme qui lit A, B et C au clavier et affiche les éventuelles solutions après s'être assuré que l'équation est bien du second degré.

**Réponse**

Programme SOLUTION

Variable A, B, C, $X_1$, $X_2$, $X_0$, S, Δ : réels

début
lire (S)
Si Δ > 0
écrire $X_1 \leftarrow \frac{-B-\sqrt{\Delta}}{2A}$ et $X_2 \leftarrow \frac{-B+\sqrt{\Delta}}{2A}$
sinon
écrire $X_0 \leftarrow \frac{-B}{2A}$
fSi

**Exercice N°3**

Écrire un algorithme qui calcule puis affiche tous les carrés des entiers inférieurs ou équivalents à 10.

**Réponse**

```
Programme MULTIPLICATION
Variable J réels
répété J ≥ 10
début
  lire (J)
J ← J+1
  écrire J ← J²
  fin
```

**Exercice N°4**

Écrire un algorithme qui calcule le Plus Grand Commun Diviseur (PGCD) de deux entiers naturels par la méthode des différences successives.

**Réponse**

```
Programme PGCD
Variable S, C, D, E, M : réels
  début
    lire (S)
    écrire D/M ← S*E/S*C
  fin
```

**ANNEXE 3 :** Test proposé au post-test

**ALGORITHMIQUE**
**LICENCE 1 – STI**
**2 heures (documents autorisés)**

**Exercice N°1**

Écrire un algorithme qui lit un nombre entier et vous affiche après vérification si ce nombre est pair ou impair.

*Indication : il est à rappeler qu'un nombre N est pair si N* ***mod*** *2 est égal à 0*

**Réponse**

**Exercice N°2**

Soit une équation du second degré de la forme $Ax^2 + Bx + C = 0$.

Écrire un algorithme qui lit A, B et C au clavier et affiche les éventuelles solutions après s'être assuré que l'équation est bien du second degré.

*Indication : $ax^2 + bx + c$ est du second degré si a est différent de 0. Le discriminant DELTA vaut $b^2-4ac$ ainsi si DELTA est >0, on a une racine double $x = \frac{-b \pm \sqrt{b^2-4ac}}{2a}$ et sinon pas de solutions réelles.*

**Réponse**

**Exercice N°3**

Écrire un algorithme qui calcule puis affiche tous les carrés des entiers inférieurs ou équivalents à 10.

*Indication : on désigne par **J** le carré de l'entier **I** et on note **J = I x I** tant que **I** reste inférieur ou équivalent à 10.*

**Réponse**

**Exercice N°4**

Écrire un algorithme qui calcule le Plus Grand Commun Diviseur (PGCD) de deux entiers naturels par la méthode des différences successives.

*Indication : pour déterminer le PGCD de deux entiers naturels, on peut les remplacer **successivement** par **le plus petit des deux de leur différence**. La dernière différence non nulle est le PGCD de ces deux nombres*

**Réponse**

**ANNEXE 4 :** Exemple de production d'un apprenant au post test

S.25 9/20

**ALGORITHMIQUE**
**LICENCE 1 – STI**
**2 heures (documents [redacted] autorisés)**

**Exercice N°1**
Écrire un algorithme qui lit un nombre entier et vous affiche après vérification si ce nombre est pair ou impair.
*Indication : il est à rappeler qu'un nombre N est pair si N **mod** 2 est égal à 0*

**Réponse**

0

**Exercice N°2**
Soit une équation du second degré de la forme $Ax^2 + Bx + C = 0$.
Écrire un algorithme qui lit A, B et C au clavier et affiche les éventuelles solutions après s'être assuré que l'équation est bien du second degré.
*Indication : $ax^2 + bx + c$ est du second degré si a est différent de 0. Le discriminant DELTA vaut $b^2-4ac$ ainsi si DELTA est >0, on a une racine double $x = \frac{-b \pm \sqrt{b^2-4ac}}{2a}$ et sinon pas de solutions réelles.*

**Réponse**

Programme LIRE ET AFFICHE LES SOLUTIONS

Variables
A, B, C, X, Δ : réels

début
Lire (A, B, C)
a * Sqrt(b) ← (a != 0)
Lire (Δ)
Srqb-4*a*c ← Δ
Ecrire (Srqb-4*a*c)
Si Srqb - 4*a*c > 0 alors
X ← (-b ± Srqt(Srqb-4*a*c))/(2*a)
Ecrire (X) (Affiche $x_1$ ; $x_2$)
fin si
Si non (Affiche pas de solution)
fin si

3/5

**Exercice N°3**

Écrire un algorithme qui calcule puis affiche tous les carrés des entiers inférieurs ou équivalents à 10.

*Indication : on désigne par **J** le carré de l'entier **I** et on note **J = I x I** tant que **I** reste inférieur ou équivalent à 10.*

**Réponse**

```
Programme CALCUL TOUT LES CARRÉS
Variable
  I, J : entiers
debut
lire(I)
Si I ≠ 0 alors
  J ← I*I
  Ecrire(J) affiche(0).
Si (I≠1) alors
  J ← I*I
  Ecrire(J) affiche(1)
Si I+1
  I ← I=2
  J ← I*I
  Ecrire(J) affiche(4)
Si I = 3 alors
  J ← I*I
  Ecrire(J) affiche(9)
Si I=4 alors
  J ← I*I
  Ecrire(J) affiche(16)
Si I = 5 alors
```

**Exercice N°4**

Écrire un algorithme qui calcule le Plus Grand Commun Diviseur (PGCD) de deux entiers naturels par la méthode des différences successives.

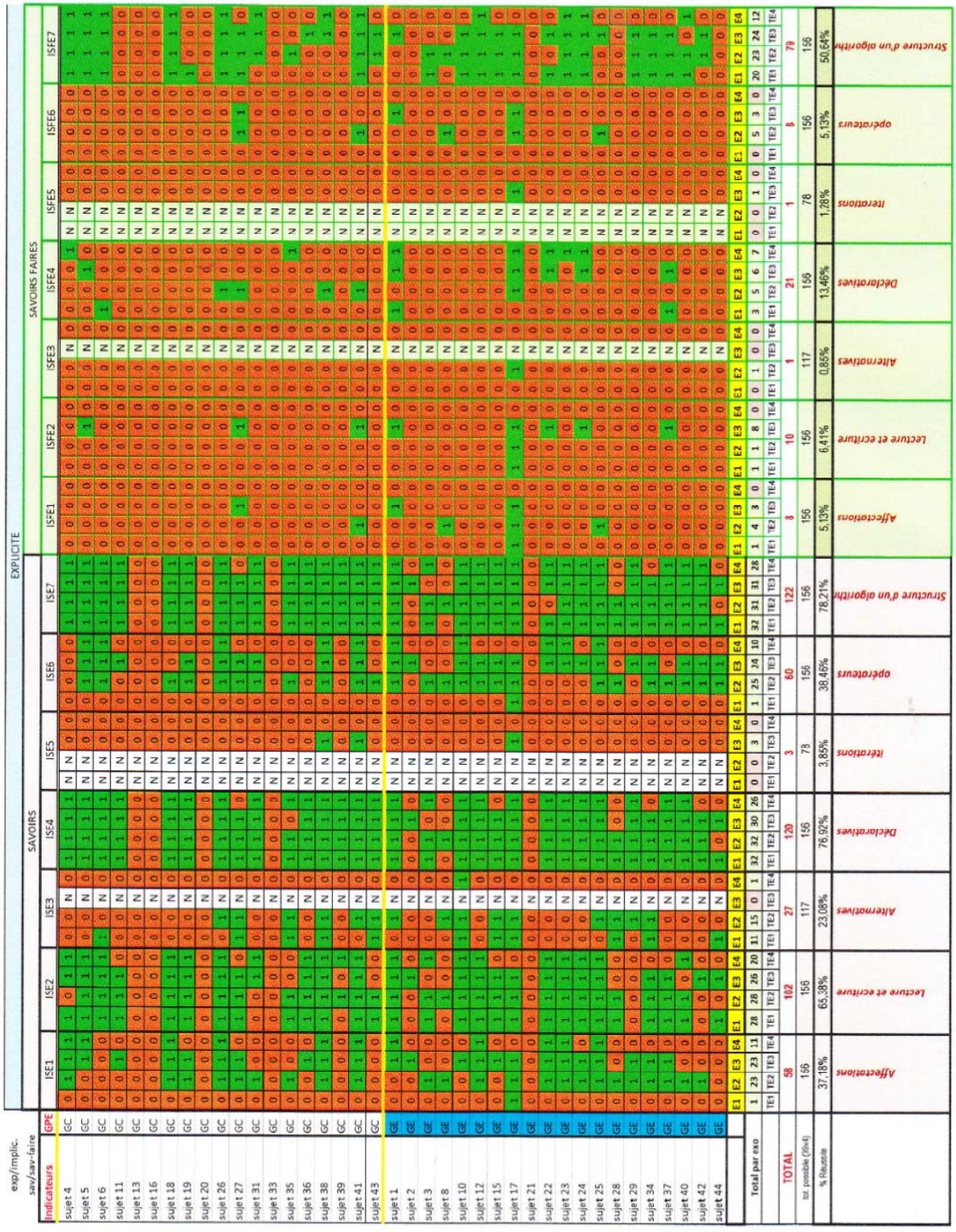
*Indication : pour déterminer le PGCD de deux entiers naturels, on peut les remplacer **successivement** par **le plus petit des deux de leur différence**. La dernière différence non nulle est le PGCD de ces deux nombres*

**Réponse**

```
Programme CALCUL DU PGCD
Variables
  A, B, S : entiers
debut
lire(A,B)
tant que A > B faire
  S ← A/B
  Ecrire(S)
pour S1 > B faire
  S ← S1/B
```

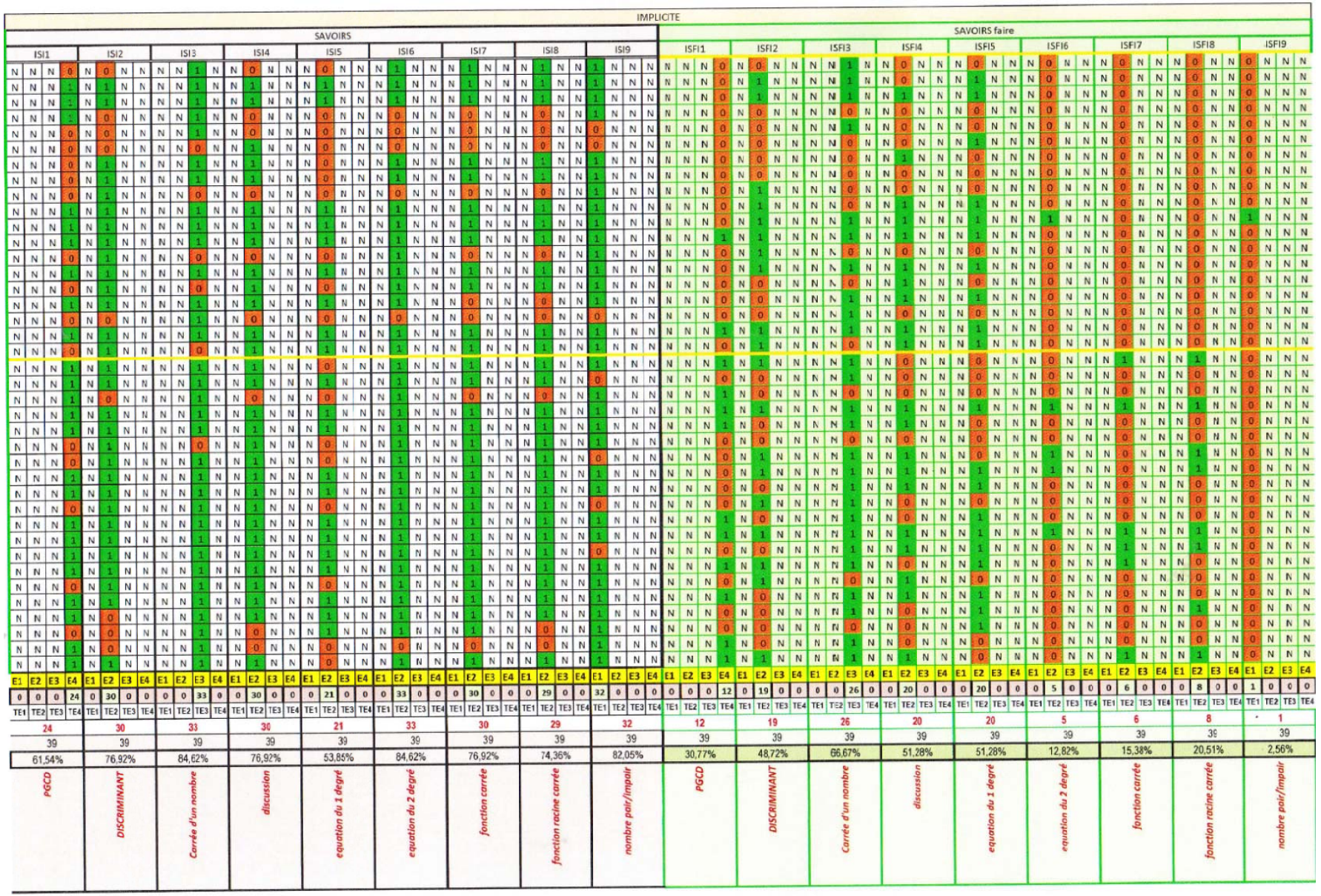
**ANNEXE 5 :** Tableau de codification des productions des apprenants

## 1- Au pré-test




IMPLICITE
SAVOIRS
SAVOIRS faire
ISI1
ISI2
ISI3
ISI4
ISI5
ISI6
ISI7
ISI8
ISI9
ISF1
ISF2
ISF3
ISF4
ISF5
ISF6
ISF7
ISF8
ISF9
PGCD
DISCRIMINANT
Carrée d'un nombre
discussion
equation du 1 degré
equation du 2 degré
fonction carrée
fonction racine carrée
nombre pair/impair

## 2- Au Post test


IMPLICITE
SAVOIRS
SAVOIRS faire
ISI1
ISI2
ISI3
ISI4
ISI5
ISI6
ISI7
ISI8
ISI9
ISFI1
ISFI2
ISFI3
ISFI4
ISFI5
ISFI6
ISFI7
ISFI8
ISFI9
PGCD
DISCRIMINANT
Carrée d'un nombre
discussion
equation du 1 degré
equation du 2 degré
fonction carrée
fonction racine carrée
nombre pair/impair

**ANNEXE 6 :** Liste des savoirs explicites et implicites par exercice

## 1- Savoirs et savoir-faire explicites

| SAVOIRS | SAVOIR-FAIRE | |
|---|---|---|
| SE1 | | |
| attribuer à une variable une valeur | S<----A +B | SFE1 |
| SE2 | | |
| afficher à l'écran une variable | ***écrire***(A) | SFE2 |
| SE3 | | |
| afficher à l'écran un texte | ***écrire***("la valeur de A est") | SFE3 |
| SE4 | | |
| Affichage texte et variable | ***écrire***("la valeur de A est de:", A) | SFE4 |
| SE5 | | |
| Alternative double | ***si*** condition ***alors*** ....<br>***sinon*** ...<br>***Fsi*** | SFE5 |
| SE6 | | |
| alternative multiple | ***selon que*** condition 1 ***faire*** ...<br>***ou que*** condition 2 ***faire*** ....<br>***ou que*** condition 3 ***faire*** ....<br>...<br>***autrement faire*** ...<br>***Fselon*** | SFE6 |
| SE7 | | |
| alternative simple | ***si*** condition ***alors*** ....<br>***Fsi*** | SFE7 |
| SE8 | | |
| déclaration d'une variable entière | A: entier | SFE8 |
| SE9 | | |
| déclaration d'une variable réelle | A: réel | SFE9 |
| SE10 | | |
| fonction prédéfinie (racine carrée) | R <---- ***sqrt(***D***)*** | SFE10 |
| SE11 | | |
| imbrication structures alternatives | ***si*** condition ***alors*** ....<br>***si*** condition ***alors*** ....<br>***Fsi***<br>***Fsi*** | SFE11 |
| SE12 | | |
| imbrication structures itératives | ***tantque*** condition ***faire***....<br>***tantque*** condition ***faire***....<br>***Ftant***<br>***Ftant*** | SFE12 |
| SE13 | | |
| imbrication itération et alternative | ***tantque*** condition ***faire***....<br>***si*** condition ***alors***....<br>***Fsi*** | SFE13 |

| | | |
|---|---|---|
| | ***Ftant*** | |
| SE14 | | |
| incrémentation | R <---- R+1 | SFE14 |
| SE15 | | |
| initialisation | R <---- 0 | SFE15 |
| SE16 | | |
| lecture de variables | lire(A) | SFE16 |
| SE17 | | |
| opérateur arithmétique (modulo) | R <---- D ***mod*** d | SFE17 |
| SE18 | | |
| opérateurs relationnels | A **<>** B<br>A < = B<br>A **=** B | SFE18 |
| SE19 | | |
| opérateurs arithmétiques | A * B / 2 | SFE19 |
| SE20 | | |
| structure itérative | ***tantque*** condition ***faire....***<br>***Ftant*** | SFE20 |
| SE21 | | |
| Structure d'un algorithme | **Programme**<br>**....**<br>**Début**<br>**.....**<br>**Fin** | **SFE21** |

## 2- Savoirs et savoir-faire implicites

| | SAVOIRS | SAVOIRS FAIRE | |
|---|---|---|---|
| | SI2 | | |
| S1 | calcul du discriminant | D <----- sqr(B) - 4 * A * C | SFI2 |
| | SI2 | | |
| S2 | carré d'un nombre | ***A * A***<br>***Sqr(A)*** | SFI2 |
| | | | |
| S4 | équation du premier degré | ***lire (A, B, C)***<br>***....***<br>***si A=0*** | SFi5 |
| | SI5 | | |
| S5 | équation du second degré | ***lire (A, B, C)***<br>***si*** A <> 0 ***alors*** .... | SFI5 |
| | SI3 | | |
| S3 | discussion en fonction du discriminant | ***si*** D < 0 alors ...<br>***si*** D = 0 alors ....<br>***si*** D > 0 alors .... | SFI3 |
| | SI7 | | |
| S6 | forme des deux racines | **$X_1$ <---- -B - sqrt(D) / (2 * A)**<br>**$X_2$ <---- -B + sqrt(D) / (2 * A)** | SFI6 |

| | | | |
|---|---|---|---|
| | SI8 | | |
| S7 | forme de la racine double | **X0 <---- -B / (2 * A)** | SFI7 |
| | SI9 | | |
| S8 | la fonction carrée | sqr() | SFI8 |
| | SI10 | | |
| S10 | nombre impair | 1 <---- D mod 2 | SFI10 |
| | SI9 | | |
| S9 | méthode de calcul d'un PGCD | ***tantque*** A <> B ***faire****....*<br>***si*** A > B ***alors*** ***A<---- A-B***<br>***Fsi***<br>***Ftant*** | SFI9 |
| | SI11 | | |
| S11 | nombre pair | 0 <---- D mod 2 | SFI11 |
| | SI12 | | |
| S12 | notion de PGCD | aucun | SFI12 |
| | SI13 | | |
| S13 | résolution d'une équation du second degré | aucun | SFI13 |
| | SI14 | | |
| S14 | reste d'une division d'un entier par 2 | R <---- A ***mod*** 2 | SFI14 |

**ANNEXE 7 :** Grille d'observation Introversion/inhibition

| N° | ITEMS | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 1.1 | S'approprie facilement le sujet (l'énoncé) au cours de l'activité … | | | | | |
| 1.2 | Ne pose pas de question après la lecture du sujet (l'énoncé) | | | | | |
| 1.3 | Ne consulte aucune autre source d'information | | | | | |
| 2.1 | Se plait à soigner sa copie | | | | | |
| 2.2 | Très lent à rentrer dans le travail | | | | | |
| 2.3 | Reste indifférent au travail demandé | | | | | |
| 3.1 | Attitude consentante à l'invitation à se mettre au travail | | | | | |
| 3.2 | Est indifférent aux appréciations du professeur | | | | | |
| 3.3 | Ne livre jamais au professeur de déclaration sur lui-même | | | | | |
| 3.4 | Ne commente jamais ses actes, ses productions (sa solution) | | | | | |
| 4.1 | Mène sa tâche (sa production) avec un grand souci de détails peu de choses lui échappent | | | | | |
| 4.2 | Mène sa tâche (sa production) de façon étroite, reste fixée sur le sujet, sur le thème | | | | | |
| 4.3 | Écoute attentivement parler un camarade ou tient compte de ses conseils ou critiques | | | | | |
| 5.1 | Assume de façon régulière et jusqu'à la fin une petite responsabilité | | | | | |
| 5.2 | N'intervient pas dans l'élaboration de la solution collective | | | | | |
| 5.3 | Range soigneusement et spontanément ses affaires dans son cartable | | | | | |
| 6.1 | Parait indifférent au travail des autres | | | | | |
| 6.2 | Resterait de préférence en dehors du travail collectif | | | | | |
| 6.3 | Est extrêmement discret sur les actes de ses camarades | | | | | |
| 6.4 | D'humeur très stable, son comportement à l'égard d'autrui est toujours le même | | | | | |
| 7.1 | Parait maladroite et lourd | | | | | |
| 7.2 | Montre des tendances à l'isolement | | | | | |
| 7.3 | Aborde le sujet avec calme et confiance en soi | | | | | |
| 8.1 | N'a jamais eu de crises de colères ou de nerfs, ne se laisse pas aller à des insolences. | | | | | |
| 8.2 | Commence à percevoir la nécessité d'une règle syntaxique utilisée dans la solution | | | | | |
| 8.3 | Refuse toute situation agressive soit en se dérobant, soit en cherchant la conciliation | | | | | |
| 9.1 | Ne prend jamais d'initiative. | | | | | |
| 9.2 | Est indifférent aux reproches et critiques de ses commandes | | | | | |
| 9.3 | Refuse toute situation agressive soit en se dérobant, soit en cherchant la conciliation | | | | | |
| 9.4 | Il se dérobe d'une épreuve de force avec ses camarades | | | | | |

| | Introversion / Inhibition | Résultats tous les sujets | | | | |
|---|---|---|---|---|---|---|
| **N°** | **ITEMS** | **1** | **2** | **3** | **4** | **5** |
| 1.1 | S'approprie facilement le sujet (l'énoncé) au cours de l'activité … | 13 | 13 | 5 | 14 | 11 |
| 1.2 | Ne pose pas de question après la lecture du sujet (l'énoncé) | 6 | 6 | 7 | 19 | 18 |
| 1.3 | Ne consulte aucune autre source d'information | 5 | 1 | 8 | 20 | 22 |
| 2.1 | Se plait à soigner sa copie | 5 | 5 | 3 | 19 | 24 |
| 2.2 | Très lent à rentrer dans le travail | 8 | 5 | 4 | 23 | 16 |
| 2.3 | Reste indifférent au travail demandé | 6 | 8 | 3 | 23 | 16 |
| 3.1 | Attitude consentante à l'invitation à se mettre au travail | 7 | 6 | 12 | 25 | 6 |
| 3.2 | Est indifférent aux appréciations du professeur | | | | | |
| 3.3 | Ne livre jamais au professeur de déclaration sur lui-même | | | | | |
| 3.4 | Ne commente jamais ses actes, ses productions (sa solution) | | | | | |
| 4.1 | Mène sa tâche (sa production) avec un grand souci de détails peu de choses lui échappent | 7 | 8 | 5 | 11 | 25 |
| 4.2 | Mène sa tâche (sa production) de façon étroite, reste fixée sur le sujet, sur le thème | 4 | 6 | 3 | 16 | 27 |
| 4.3 | Écoute attentivement parler un camarade ou tient compte de ses conseils ou critiques | | | | | |
| 5.1 | Assume de façon régulière et jusqu'à la fin une petite responsabilité | 3 | 7 | 1 | 25 | 20 |
| 5.2 | N'intervient pas dans l'élaboration de la solution collective | | | | | |
| 5.3 | Range soigneusement et spontanément ses affaires dans son cartable | | | | | |
| 6.1 | Parait indifférent au travail des autres | 3 | 4 | 5 | 32 | 12 |
| 6.2 | Resterait de préférence en dehors du travail collectif | | | | | |
| 6.3 | Est extrêmement discret sur les actes de ses commandes | | | | | |
| 6.4 | D'humeur très stable, son comportement à l'égard d'autrui est toujours le même | | | | | |
| 7.1 | Parait maladroite et lourd | 2 | 7 | 2 | 13 | 32 |
| 7.2 | Montre des tendances à l'isolement | 1 | 3 | 1 | 35 | 16 |
| 7.3 | Aborde le sujet avec calme et confiance en soi | 5 | 9 | 3 | 32 | 7 |
| 8.1 | N'a jamais eu de crises de colères ou de nerfs, ne se laisse pas aller à des insolences. | | | | | |
| 8.2 | Commence à percevoir la nécessité d'une règle syntaxique utilisée dans la solution | 4 | 5 | 4 | 27 | 16 |
| 8.3 | Refuse toute situation agressive soit en se dérobant, soit en cherchant la conciliation | | | | | |
| 9.1 | Ne prend jamais d'initiative. | | | | | |
| 9.2 | Est indifférent aux reproches et critiques de ses commandes | | | | | |
| 9.3 | Refuse toute situation agressive soit en se dérobant, soit en cherchant la conciliation | | | | | |
| 9.4 | Il se dérobe d'une épreuve de force avec ses camarades | | | | | |
| | | **79** | **93** | **66** | **334** | **268** |

**ANNEXE 8 :** Grille d'observation Introversion/inhibition (complétée)

## Introversion / Inhibition

| N° | ITEMS | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 1.1 | S'approprie facilement le sujet (l'énoncé) au cours de l'activité … | | X | | | |
| 1.2 | Ne pose pas de question après la lecture du sujet (l'énoncé) | | | | X | |
| 1.3 | Ne consulte aucune autre source d'information | | | | X | |
| 2.1 | Se plait à soigner sa copie | | | | X | |
| 2.2 | Très lent à rentrer dans le travail | | X | | | |
| 2.3 | Reste indifférent au travail demandé | | X | | | |
| 3.1 | Attitude consentante à l'invitation à se mettre au travail | | | | X | |
| 3.2 | Est indifférent aux appréciations du professeur | | | | | |
| 3.3 | Ne livre jamais au professeur de déclaration sur lui-même | | | | | |
| 3.4 | Ne commente jamais ses actes, ses productions (sa solution) | | | | | |
| 4.1 | Mène sa tâche (sa production) avec un grand souci de détails peu de choses lui échappent | | | | X | |
| 4.2 | Mène sa tâche (sa production) de façon étroite, reste fixée sur le sujet, sur le thème | | | | X | |
| 4.3 | Ecoute attentivement parler un camarade ou tient compte de ses conseils ou critiques | | | X | | |
| 5.1 | Assume de façon régulière et jusqu'à la fin une petite responsabilité | | | X | | |
| 5.2 | N'intervient pas dans l'élaboration de la solution collective | | | | | |
| 5.3 | Range soigneusement et spontanément ses affaires dans son cartable | | | | | |
| 6.1 | Parait indifférent au travail des autres | | | | X | |
| 6.2 | Resterait de préférence en dehors du travail collectif | | | | | |
| 6.3 | Est extrêmement discret sur les actes de ses commandes | | | | | |
| 6.4 | D'humeur très stable, son comportement à l'égard d'autrui est toujours le même | | | | | |
| 7.1 | Parait maladroite et lourd | X | | | | |
| 7.2 | Montre des tendances à l'isolement | X | | | | |
| 7.3 | Aborde le sujet avec calme et confiance en soi | | | | X | |
| 8.1 | N'a jamais eu de crises de colères ou de nerfs, ne se laisse pas aller à des insolences. | | | | | |
| 8.2 | Commence à percevoir la nécessité d'une règle syntaxique utilisée dans la solution | | | | X | |
| 8.3 | Refuse toute situation agressive soit en se dérobant, soit en cherchant la conciliation | | | | | |
| 9.1 | Ne prend jamais d'initiative. | | | | | |
| 9.2 | Est indifférent aux reproches et critiques de ses commandes | | | | | |
| 9.3 | Refuse toute situation agressive soit en se dérobant, soit en cherchant la conciliation | | | | | |
| 9.4 | Il se dérobe d'une épreuve de force avec ses camarades | | | | | |

**ANNEXE 9 :** Grille d'observation Excitabilité/instabilité/extraversion

| N° | ITEMS | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 12.1 | S'agite constamment ou manipule des objets au cours de l'activité | | | | | |
| 12.2 | Pose très fréquemment des questions après la lecture du sujet | | | | | |
| 12.3 | Consulte d'autres sources d'information | | | | | |
| 22.1 | N'accorde aucun soin à sa copie. | | | | | |
| 22.2 | Se met au travail rapidement | | | | | |
| 22.3 | Participe très activement au travail demandé | | | | | |
| 32.1 | Attitudes de refus à l'invitation à se mettre au travail | | | | | |
| 32.2 | Recherche constamment la valorisation de son travail par le professeur | | | | | |
| 32.3 | Se plait à des confidences, aux déclarations sincères ou trompeuses | | | | | |
| 32.4 | Éprouve très fréquemment le besoin de justifier d'expliquer ses actes, ses productions | | | | | |
| 42.1 | Mène sa tâche rapidement, se contente d'une vue globale du sujet | | | | | |
| 42.2 | Établi des similitudes avec d'autres taches, d'autres sujets, d'autres exercices. | | | | | |
| 42.3 | N'écoute pas parler un camarade …, ne tient pas compte de ses conseils ou critiques | | | | | |
| 52.1 | Assume une responsabilité au début avec enthousiasme mais abandonne très vite | | | | | |
| 52.2 | Intervient très fréquemment dans l'élaboration d'un projet collectif | | | | | |
| 52.3 | Laisse ses affaires en grand désordre dans son cartable | | | | | |
| 62.1 | S'occupe beaucoup du travail des autres, soit en les aidant, soit en les critiquant | | | | | |
| 62.2 | Prend facilement l'initiative d'organiser le travail collectif | | | | | |
| 62.3 | Très forte tendance à dénoncer ses camarades | | | | | |
| 62.4 | D'humeur très habile, comportement changeant à l'endroit d'autrui | | | | | |
| 72.1 | Parait adroit et vif | | | | | |
| 72.2 | Est toujours partie prenante dans les groupes les plus actifs | | | | | |
| 72.3 | Aborde le sujet avec excitation ou perd ses moyens à la découverte du sujet | | | | | |
| 82.1 | A toujours des crises de colères ou de nerfs en récréations, est parfois insolent | | | | | |
| 82.2 | Est très loin de percevoir la nécessité d'utiliser les règles syntaxiques dans la solution | | | | | |
| 82.3 | Accepte joyeusement l'effort physique | | | | | |
| 92.1 | Prend beaucoup d'initiative | | | | | |
| 92.2 | Est très troublés par les reproches et critiques avec ses camarades | | | | | |
| 92.3 | Est très fréquemment en état d'accrochage agressif avec ses camarades. | | | | | |
| 92.4 | Il affronte volontiers ses camarades | | | | | |

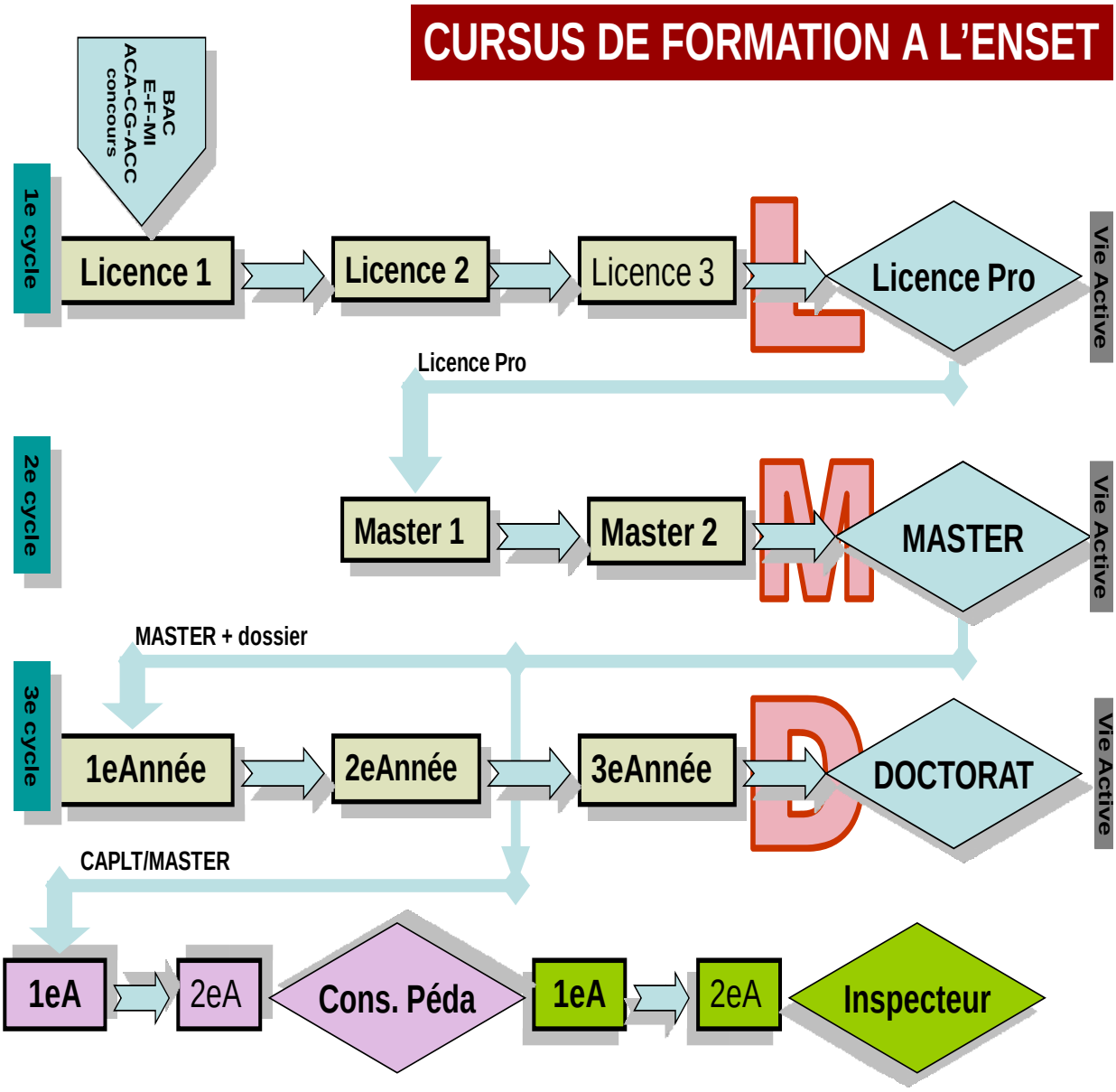
| | Excitabilité/ Instabilité/ Extraversion | Résultats tous les sujets | | | | |
|---|---|---|---|---|---|---|
| **N°** | **ITEMS** | **1** | **2** | **3** | **4** | **5** |
| 12.1 | S'agite constamment ou manipule des objets au cours de l'activité | 40 | 4 | 9 | 2 | 1 |
| 12.2 | Pose très fréquemment des questions après la lecture du sujet | 42 | 6 | 2 | 4 | 2 |
| 12.3 | Consulte d'autres sources d'information | 37 | 4 | 0 | 9 | 6 |
| 22.1 | N'accorde aucun soin à sa copie. | 15 | 14 | 13 | 7 | 7 |
| 22.2 | Se met au travail rapidement | 28 | 15 | 3 | 8 | 2 |
| 22.3 | Participe très activement au travail demandé | 23 | 22 | 2 | 8 | 1 |
| 32.1 | Attitudes de refus à l'invitation à se mettre au travail | 29 | 12 | 9 | 4 | 2 |
| 32.2 | Recherche constamment la valorisation de son travail par le professeur | | | | | |
| 32.3 | Se plait à des confidences, aux déclarations sincères ou trompeuses | | | | | |
| 32.4 | Éprouve très fréquemment le besoin de justifier d'expliquer ses actes, ses productions | | | | | |
| 42.1 | Mène sa tâche rapidement, se contente d'une vue globale du sujet | 22 | 20 | 8 | 4 | 2 |
| 42.2 | Établi des similitudes avec d'autres taches, d'autres sujets, d'autres exercices. | 31 | 21 | 2 | 1 | 1 |
| 42.3 | N'écoute pas parler un camarade …, ne tient pas compte de ses conseils ou critiques | | | | | |
| 52.1 | Assume une responsabilité au début avec enthousiasme mais abandonne très vite | 20 | 28 | 2 | 1 | 5 |
| 52.2 | Intervient très fréquemment dans l'élaboration d'un projet collectif | | | | | |
| 52.3 | Laisse ses affaires en grand désordre dans son cartable | | | | | |
| 62.1 | S'occupe beaucoup du travail des autres, soit en les aidant, soit en les critiquant | 24 | 18 | 9 | 3 | 2 |
| 62.2 | Prend facilement l'initiative d'organiser le travail collectif | | | | | |
| 62.3 | Très forte tendance à dénoncer ses camarades | | | | | |
| 62.4 | D'humeur très habile, comportement changeant à l'endroit d'autrui | | | | | |
| 72.1 | Parait adroit et vif | 19 | 29 | 5 | 2 | 1 |
| 72.2 | Est toujours partie prenante dans les groupes les plus actifs | 18 | 12 | 12 | 14 | 0 |
| 72.3 | Aborde le sujet avec excitation ou perd ses moyens à la découverte du sujet | 26 | 17 | 10 | 2 | 1 |
| 82.1 | A toujours des crises de colères ou de nerfs en récréations, est parfois insolent | | | | | |
| 82.2 | Est très loin de percevoir la nécessité d'utiliser les règles syntaxiques dans la solution | 28 | 14 | 6 | 6 | 2 |
| 82.3 | Accepte joyeusement l'effort physique | | | | | |
| 92.1 | Prend beaucoup d'initiative | | | | | |
| 92.2 | Est très troublés par les reproches et critiques avec ses camarades | | | | | |
| 92.3 | Est très fréquemment en état d'accrochage agressif avec ses camarades. | | | | | |
| 92.4 | Il affronte volontiers ses camarades | | | | | |
| | | **402** | **236** | **92** | **75** | **35** |

**ANNEXE 10 :** Grille d'observation Excitabilité/instabilité/extraversion (complétée)

## Excitabilité/ Instabilité/ Extraversion

| N° | ITEMS | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 12.1 | S'agite constamment ou manipule des objets au cours de l'activité | X | | | | |
| 12.2 | Pose très fréquemment des questions après la lecture du sujet | X | | | | |
| 12.3 | Consulte d'autres sources d'information | X | | | | |
| 22.1 | N'accorde aucun soin à sa copie. | X | | | | |
| 22.2 | Se met au travail rapidement | | | | X | |
| 22.3 | Participe très activement au travail demandé | | | | X | |
| 32.1 | Attitudes de refus à l'invitation à se mettre au travail | X | | | | |
| 32.2 | Recherche constamment la valorisation de son travail par le professeur | | | | | |
| 32.3 | Se plait à des confidences, aux déclarations sincères ou trompeuses | | | | | |
| 32.4 | Eprouve très fréquemment le besoin de justifier d'expliquer ses actes, ses productions | | | | | |
| 42.1 | Mène sa tâche rapidement, se contente d'une vue globale du sujet | | | X | | |
| 42.2 | Etabli des similitudes avec d'autres taches, d'autres sujets, d'autres exercices. | | X | | | |
| 42.3 | N'écoute pas parler un camarade ..., ne tient pas compte de ses conseils ou critiques | | | | | |
| 52.1 | Assume une responsabilité au début avec enthousiasme mais abandonne très vite | | X | | | |
| 52.2 | Intervient très fréquemment dans l'élaboration d'un projet collectif | | | | | |
| 52.3 | Laisse ses affaires en grand désordre dans son cartable | | | | | |
| 62.1 | S'occupe beaucoup du travail des autres, soit en les aidant, soit en les critiquant | X | | | | |
| 62.2 | Prend facilement l'initiative d'organiser le travail collectif | | | | | |
| 62.3 | Très forte tendance à dénoncer ses camarades | | | | | |
| 62.4 | D'humeur très habile, comportement changeant à l'endroit d'autrui | | | | | |
| 72.1 | Parait adroit et vif | | | | X | |
| 72.2 | Est toujours partie prenante dans les groupes les plus actifs | | | | X | |
| 72.3 | Aborde le sujet avec excitation ou perd ses moyens à la découverte du sujet | | | X | | |
| 82.1 | A toujours des crises de colères ou de nerfs en récréations, est parfois insolent | | | | | |
| 82.2 | Est très loin de percevoir la nécessité d'utiliser les règles syntaxiques dans la solution | X | | | | |
| 82.3 | Accepte joyeusement l'effort physique | | | | | |
| 92.1 | Prend beaucoup d'initiative | | | | | |
| 92.2 | Est très troublés par les reproches et critiques avec ses camarades | | | | | |
| 92.3 | Est très fréquemment en état d'accrochage agressif avec ses camarades. | | | | | |
| 92.4 | Il affronte volontiers ses camarades | | | | | |

**ANNEXE 11 : Cursus de formation à l'ENSET**



## ANNEXE 12 : Énoncés et propositions de solutions

- **Accomplissement d'une tâche simple**

**Énoncé N°1**

*Écrire un algorithme qui lit un nombre entier et vous affiche après vérification si ce nombre est pair ou impair.*

**Proposition Solution**

```
programme NBRE_PAIR
variables
   VALEUR, RESTE : entiers
début
   lire (VALEUR)
   RESTE ← VALEUR mod 2
      si RESTE = 0 alors
         écrire (VALEUR, "est un nombre pair ")
         sinon
         écrire (VALEUR, " est un nombre impair")
   fsi
fin
```

**Énoncé N°2**

*Écrire un algorithme qui calcule puis affiche tous les carrés des entiers inférieurs ou équivalents à 10.*

**Proposition Solution**

```
Programme CARRETANT
Variables
   I, J : entiers
début
   I ← 0
   tant que I < = 10 faire
      I ← I + 1
      J ← I * I
      écrire ( J )
   ftant
fin
```

- **Accomplissement d'une tâche complexe**

**Énoncé N°3**

*Écrire un algorithme qui calcule le Plus Grand Commun Diviseur (PGCD) de deux entiers naturels par la méthode des différences successives.*

**Proposition Solution**

```
programme PGCD
variables
   A, B : entiers
début
   lire (A, B)
   tant que A <> B faire
      si A > B alors
         A ← A – B
         Sinon
         B ← B - A
       fsi
   ftant
  écrire (B)
fin
```

**Énoncé N°4**

*Soit une équation du second degré de la forme $Ax^2 + Bx + C = 0$*

*Écrire un algorithme qui lit A, B et C au clavier et affiche les éventuelles solutions après s'être assuré que l'équation est bien du second degré.*

**Proposition Solution**

```
programme EQUATION2
variables
   A, B, C, DELTA, RAC1, RAC2 : réels
début
   lire (A, B, C)
   si  A <> 0 alors
      DELTA ← sqr(B) – 4 * A * C
         si  DELTA  =  0 alors
            RAC1 ← - B / (2*A)
            écrire ( "RAC1 = " , RAC1)
         fsi
         si DELTA > 0  alors
            RAC1 ←  (- B -  sqrt(DELTA) / (2 * A))
            RAC2 ←  (- B + sqrt(DELTA) / (2 * A))
```

```
        écrire ( " RAC1 =", RAC1, " RAC2 =", RAC2)
      fsi
      si DELTA < 0 alors
        écrire ("pas de solution réelle ")
      fsi
    sinon
     écrire (" ce n'est pas une équation de second degré")
  fsi
fin
```