|  |
| --- |
| **Informatique et enseignement général**  par **Jacques Arsac** Directeur de la section informatique École normale supérieure, Membre correspondant de l'académie des Sciences  **1. Informatisation de la société**     Nous voulons placer ce rapport dans la perspective de l'informatisation de la société. C'est un fait, reconnu et encouragé par le gouvernement. Elle pose un certain nombre de problèmes, dont trois seront retenus ici.  ***1.1. Information du citoyen***     L'informatique est perçue le plus souvent comme un phénomène technique, la construction et l'utilisation des ordinateurs. Ce n'est que la partie visible de l'iceberg. Mais cela pose déjà de nombreux problèmes. Les emplois s'en trouvent affectés : création de nouveaux emplois en informatique, mais, on le craint aussi, fermeture d'autres emplois comme conséquence de l'automatisation. Modification des conditions de travail ; l'emploi de terminaux connectés par téléphone ne va-t-il pas ramener le travail à domicile ? Modification des conditions de vie : le terminal à domicile, la carte de crédit généralisée, l'organisation des loisirs... Le citoyen trouvera l'informatique partout.     Tout ceci se déroule dans un contexte où l'informatique est auréolée de mystère en raison de l'extrême complexité des ordinateurs que l'on s'est complu à présenter comme des « cerveaux artificiels ». Dès lors on peut craindre des réactions de rejet, comme cela s'est déjà passé pour la plupart des grandes inventions techniques ou scientifiques. Les récentes attaques d'ordinateurs en sont une des manifestations. La réponse à la superstition, c'est le savoir. Le citoyen a le droit, et peut-être le devoir, de savoir ce qui se passe, ce que recouvre l'informatique, ce qu'est l'information qu'elle manipule... L'informatisation de la société doit s'appuyer sur un effort de formation.  ***1.2. Ouverture sur les professions***     L'élève de lycée sera le professionnel ou l'utilisateur de demain. C'est du lycée que sortiront les 20 000 candidats aux emplois dont on prévoit qu'ils seront créés par l'informatique chaque année. Il est souhaitable qu'un premier contact avec cette science ait lieu dès le lycée, pour favoriser l'éclosion des vocations. Certes tous les lycéens ne sont pas concernés par l'informatique aussi directement. Mais nous insisterons sur le fait que cette science a des valeurs culturelles ; même si le lycéen ne doit jamais plus programmer, il lui restera quelque chose de ce qu'il aura appris au lycée, lorsqu'il aura oublié tous les aspects techniques contingents.  ***1.3. Informatisation de l'école***     La décision d'implanter 10 000 micro-ordinateurs dans les écoles secondaires est une manifestation de l'informatisation de la société. L'école ne peut rester à part. Mais ceci peut s'entendre de deux façons.  *1.3.1. L'informatique au service de l'école*     L'outil informatique a sa place parmi les outils pédagogiques modernes. Les différentes disciplines peuvent y faire appel comme aide à l'enseignement : simulation d'expériences, ou dépouillement de travaux pratiques de physique, analyse textuelle en lettres, conjugaison latine, étude de phénomènes biologiques par simulation... Ceci fait depuis 10 ans l'objet d'une expérimentation en France. Nous y reviendrons.     Il est assez raisonnable de penser que ces ordinateurs serviront aussi à la gestion de l'école. Ce sera une autre occasion de contact du citoyen avec l'informatique. Il ne serait pas normal qu'on ne s'appuie pas sur ces expériences concrètes.  *1.3.2. L'enseignement de l'informatique*     Nous pensons que l'informatique a sa place à l'école au milieu des autres disciplines scientifiques : mathématiques, physique-chimie, biologie, en raison de sa spécificité, de l'originalité de ses méthodes, et de l'extraordinaire enrichissement de la pensée scientifique qui en est résulté. La science informatique est une composante essentielle de la culture scientifique de la fin du XXe siècle. Il serait anormal qu'à l'heure de l'informatisation de la société, cette conquête de l'esprit soit passée sous silence auprès des jeunes.     Il y a là deux visions de l'informatique au lycée, que l'on doit distinguer soigneusement, mais que l'on peut concevoir comme complémentaires ou antinomiques, et dont il faut mesurer la valeur et les chances de succès.  **2. Le colloque de Sèvres**     Dès 1970, la communauté scientifique internationale se préoccupait du problème de l'introduction de l'informatique dans l'enseignement du second degré, et, sous l'égide de l'OCDE, un colloque sur ce thème se tenait à Sèvres. Comme il y est fait souvent référence dans les rapports traitant de ce sujet, nous en redonnons ici les conclusions et recommandations, dans leurs grandes lignes. Nous ne ferons pas mention des interventions de tel ou tel, qui n'engagent que leur auteur, pour ne parler que des textes élaborés par l'ensemble des participants, et qui sont significatifs de ce qui était tenu pour important en 1970.     Les conclusions reconnaissent l'intérêt de l'ordinateur comme auxiliaire pédagogique. Nouveau venu enfin dans la gamme des moyens pédagogiques, l'ordinateur est utilisé comme auxiliaire d'enseignement en général (*L'enseignement de l'informatique à l'école secondaire*, publications de l'OCDE, Paris 1971, page 34). Et aussi : l'avènement de l'ordinateur influence de nombreuses disciplines. Les applications de l'ordinateur à toute discipline où elles sont justifiées devraient être développées de préférence dans le cadre de cette discipline. Les contacts entre les enseignants chargés de cette dernière et ceux compétents dans le domaine de l'informatique devraient être facilités (id. page 43).     Ceci étant noté, la plus longue part des conclusions et recommandations est consacrée à l'enseignement de l'informatique. Dès l'ouverture des conclusions, le colloque affirme : « L'accord a été général parmi les participants au Séminaire pour affirmer que ce qui était important dans cette introduction était *non pas* l'ordinateur, *mais bien la démarche informatique que l'on peut caractériser comme algorithmique, opérationnelle, organisationnelle.* À ce titre, on peut affirmer que l'informatique, et son enseignement à ce niveau, est un moyen et non une fin en soi. Elle est un langage permettant de décrire et de comprendre certains des aspects du monde qui nous entoure » (id. page 33). La partie en italique était soulignée dans le texte. Rappelons que l'unanimité ici évoquée concernait pour la France MM. Arsac, Boussard, Hebenstreit, Jaulin, Lumbroso, Poly, Poulain.     Cette idée est reprise dans le paragraphe suivant : « L'informatique est avant tout un langage, un système de signes qui permet de communiquer au même titre que d'autres langages, tels que les mathématiques ou les langues. Elle possède, atout majeur, mais aussi contrainte formatrice, la rigueur nécessaire à une approche scientifique » (id. page 33).     Les conclusions parlent ensuite des méthodes et moyens. « S'il est vrai que le contenu de cette introduction à l'informatique dépend partiellement de la structure de l'enseignement (...) et des conditions locales (...), le Séminaire a été unanime à souligner l'intérêt qu'il y a à mettre l'élève en contact avec l'ordinateur, plus généralement avec les systèmes de traitement de l'information. Dans la mesure où le contenu de cet enseignement d'introduction prévoit une initiation à la programmation conduisant à la rédaction de programmes par les élèves, il est nécessaire, au moins pour des raisons pédagogiques, que ces derniers voient le résultat de leur travail et prennent conscience ainsi de ce qu'un ordinateur est en mesure de faire, et aussi de ce qu'il exige comme rigueur et exactitude » (id. page 34).     Les conclusions abordent le problème de la formation des enseignants, considérant que « l'enseignement de l'informatique est une nécessité à laquelle les responsables de l'éducation n'échapperont pas » (id. page 35). Elles insistent sur le fait que « on ne devrait pas se contenter de faire appel aux professeurs de mathématiques, quand bien même ces derniers, dans les expériences en cours, constitueraient le noyau initial des *enseignants d'informatique* » (page 35).     « D'une part il faut se garder de la facilité qui consisterait à se contenter d'un enseignement superficiel purement descriptif ne mettant pas en valeur, concrètement, le caractère privilégié de l'informatique (dans sa démarche, ses moyens d'analyse, ses outils, etc.} de l'autre, il faut éviter de donner à l'informatique plus d'importance qu'elle ne peut en avoir, tout spécialement à ce niveau » (id. page 37).     Signalons enfin cette phrase des conclusions : « développer les contacts entre l'enseignement de l'informatique et l'enseignement des autres disciplines » (page 39) car elle n'a de sens que s'il existe effectivement un enseignement d'informatique en tant que tel.     De la même façon, dans les recommandations du groupe de travail n° 1 (page 43). « Les applications de l'ordinateur à toute discipline où elles sont justifiées devraient être développées de préférence dans le cadre même de cette discipline. Les contacts entre les enseignants de cette dernière et ceux compétents dans le domaine de l'informatique devraient être facilités » (page 43). Ceci n'a de sens que s'il existe effectivement des enseignants spécialement chargés de l'informatique.     Le groupe de travail n° 2 a concentré ses efforts sur le cours d'initiation » (page 45), « II a estimé que, pour des raisons pédagogiques, ces cours d'initiation devraient inclure des travaux pratiques comprenant des passages sur ordinateur de programmes établis par les élèves » (page 45).     Nous avons peut-être trop insisté sur les conclusions du colloque de Sèvres. Il nous a paru important de rétablir les faits avec soin, des citations abusives ayant trop souvent déformé l'esprit des conclusions de ce colloque.  **3. L'expérience française**     Pour sa présentation, nous nous référons au texte que Wladimir Mercouroff présentait au colloque « Informatique et enseignement » tenu à Marseille en 1975, sous l'égide de l'IFIP (fédération internationale pour le traitement de l'information). Il présente la création d'une commission d'étude, dont « les conceptions générales ont inspiré des mesures importantes prises en 1970 et les années suivantes par le ministère de l'Éducation nationale pour introduire un enseignement de sensibilisation à l'informatique dans les établissements d'enseignement secondaire » (comptes rendus du colloque, North-Holland publishing company. p. 779). Cette idée d'un enseignement se retrouve un peu plus loin : « Ainsi, à côté des besoins en spécialistes, il est apparu nécessaire d'une part d'introduire une large *initiation* à l'informatique pour donner accès à l'outil à tous ceux qui auraient à se servir de l'informatique dans leur activité professionnelle ultérieure, sans pour autant être informaticiens ; d'autre part, ces formations devraient s'appuyer sur une large sensibilisation à l'informatique, destinée à contribuer à la culture générale des citoyens, et à préparer les esprits à l'introduction de l'informatique dans un nombre sans cesse croissant d'activités humaines d'un pays moderne ».     Cette culture générale, W. Mercouroff la présente plus loin : « Elle a pour but, non pas d'apprendre l'informatique, mais d'apprendre que l'informatique existe, à quoi elle peut servir, ce qu'elle ne peut pas faire, quelles sont ses limites, quels sont les aspects économiques qui sont associés (...) Le rôle de culture générale de l'informatique avait déjà été souligné par un colloque OCDE qui s'est tenu à Sèvres en 1970, et qui avait préconisé l'introduction de l'informatique dans l'enseignement secondaire ».     On notera toutefois la différence de contenu : alors que le colloque de Sèvres insiste sur le contenu de cet enseignement et sur les aptitudes qu'il doit développer, mentionnant les travaux pratiques de programmation, l'interprétation de W. Mercouroff restreint cet enseignement à un rôle descriptif ne pénétrant pas dans l'informatique elle-même. Il faudrait citer tout le texte de W. Mercouroff. « On aurait enseigné l'informatique en tant que telle, en décrivant aux élèves ce qu'est un ordinateur, quelles sont ses possibilités, et partant de là, en leur apprenant à communiquer avec la machine à travers un langage de programmation. Cette orientation, conforme à une certaine conception de l'informatique comme science des ordinateurs, a d'ailleurs souvent été choisie par les pays qui ont amorcé un enseignement de l'informatique à ce niveau. D'un autre côté, on peut partir de l'idée que l'informatique n'est pas la science des ordinateurs, qu'il existe une science informatique caractérisée non par les possibilités des machines, mais par des principes propres à la représentation et à la manipulation de l'information, pour ne déboucher qu'ensuite sur son traitement. C'est cette seconde optique qui a été retenue, avec pour conséquence que l'informatique ne devait pas être enseignée dans l'enseignement secondaire général comme discipline autonome, mais au niveau de toutes les disciplines traditionnelles comme méthode de raisonnement et d'analyse, en illustrant l'apport qu'elle pouvait avoir tant dans ses démarches intellectuelles (algorithmique, organigrammes...) que par ses méthodes {analyse des problèmes, structuration des données...) ou par ses techniques (traitement sur ordinateur) ».     Nous laissons à W. Mercouroff la responsabilité de cette argumentation. Nous partageons d'autant plus ses distinctions entre informatique et science des ordinateurs que nous avons été parmi les premiers à les présenter (colloque de Sèvres, 1970, premier colloque IFIP sur l'enseignement de l'informatique, Amsterdam 1970, ouvrage *La science informatique*, Paris, Dunod 1970). Mais nous voyons mal les conclusions qui en sont tirées. S'agissant de culture générale et sensibilisation à l'informatique, W. Mercouroff demande que soit présentée l'informatique, ce qu'elle peut faire, ses limites. Mais par ailleurs, il ne faut pas enseigner ce qu'est un ordinateur, quelles sont ses possibilités...     Le colloque de Sèvres avait milité fortement pour un enseignement de l'informatique, allant jusqu'à des travaux pratiques. Il avait laissé ouverte la question de savoir s'il fallait en faire un enseignement autonome, ou le placer dans le contexte d'une autre discipline. L'optique est ici différente. Il n'est plus question de parler d'ordinateurs ou de langage de programmation, mais les organigrammes et la structuration des données sont explicitement mentionnés.     En mai 1980, nous n'avons à notre disposition aucun document officiel permettant d'évaluer le résultat de 10 ans d'une telle expérience dans 58 lycées. Mais nous avons quelques jalons.     Cette expérience invitait les professeurs de toutes disciplines à utiliser l'ordinateur comme outil pédagogique à l'intérieur de leur propre discipline. À ce niveau, l'expérience est une parfaite réussite. Les professeurs de lycée français ont présenté les résultats de leurs travaux au colloque de Marseille, qu'il s'agisse des lettres : emploi d'index pour l'étude de textes (D. Valentin et J. Prévot), de la conjugaison latine (P. Muller), ou d'enquête sociologique (A. Baro et R. Fruhauf) ; de biologie (P. Aboudarham et C. Lafond) ; de mathématiques (G. Lopata) ou de probabilités (C. Helmstetter et G. Thomas) ; de simulation en physique (M. Montredon) ou de dépouillement des travaux pratiques (J. Arsac)... Cette liste n'est pas complète. Les professeurs de lycée français ont apporté une contribution massive et remarquée à ce colloque, non par la qualité informatique de leurs travaux (on a vu de meilleurs programmes} mais par la valeur pédagogique des applications ainsi développées.     Mais l'expérience avait une autre ambition, clairement marquée dans la présentation de W. Mercouroff : être un enseignement de sensibilisation et de culture générale en informatique. Il nous est impossible de dire si elle a atteint cet objectif, et ce que les élèves sortant des 58 lycées informatisés savent de cette nouvelle science, jusqu'à quel point ils en ont perçu les méthodes d'analyse, les démarches algorithmiques. Ont-ils entendu parler d'organigrammes ou de structuration de données ?     Si l'on replace l'expérience en face des objectifs du colloque de Sèvres, force est de reconnaître que nous sommes loin de ce qui était recommandé en 1970.     J. Hebenstreit a présenté au colloque de Sèvres cette fameuse expérience française. On ne trouve que dans son exposé l'idée que l'enseignement demandé par ce Séminaire serait obtenu parce que les professeurs des différentes disciplines, formés à l'informatique, développeraient des applications dans le cadre de leur propre discipline (ouvrage cité, page 219). Or dans un rapport récent, il déclare : « À les en croire, l'exercice de la programmation apprendrait aux élèves :  – à penser logiquement  – à formuler les solutions de façon claire et exhaustive  – à ne négliger aucun détail  – etc.  Objectivement, les tenants de cette thèse prennent leurs désirs pour des réalités ».     Si donc un enseignement de programmation ne peut atteindre ces objectifs, qui étaient les fondements sur lesquels s'appuyait le séminaire de Sèvres, alors il ne faut guère compter qu'une sensibilisation beaucoup plus légère comme W. Mercouroff la présente ait pu les atteindre. Si tel avait été le cas, la réalité serait là pour répondre aux désirs...     Il ne nous paraît pas évident que l'expérience française, réussite en ce qui concerne l'emploi des ressources informatiques dans l'enseignement du second degré, ait atteint l'objectif second qu'elle s'était fixé : sensibiliser les élèves aux méthodes de l'informatique, et encore moins celui du séminaire de Sèvres : développer des aptitudes algorithmiques, opérationnelles, organisatrices.     On nous objectera que dans les lycées informatisés se sont développés des clubs informatiques où des élèves ont appris un langage de programmation, écrit des programmes, assuré leur mise au point sur ordinateurs. J. Hebenstreit lui-même marque les limites du phénomène « les clubs informatiques sont des phénomènes intéressants, mais ils ne sont pas, pour l'enseignement, plus significatifs que les clubs d'astronomie ou les clubs de fusée ».     Un autre danger menace ces clubs. Les élèves y reçoivent une formation sommaire, le plus souvent réduite à la description du langage LSE. Après quoi, ils s'essaient à l'écriture de programmes. C'est du reste ce qui ressort clairement de l'article paru dans le n° 1 de la revue *Éducation et informatique* (Nathan, avril 1980), Faut-il alors parler de formation ?     Ainsi, au terme de cette analyse, il nous paraît conforme à la réalité de poser les conclusions suivantes : - en 1970, la communauté scientifique internationale recommandait l'introduction d'un enseignement d'informatique dans les lycées ; - la France a choisi une expérience de formation des maîtres de toutes disciplines à l'informatique ; - elle a favorisé l'emploi de l'ordinateur comme ressource pédagogique obtenant des résultats remarquables sur la scène internationale ; - la question de savoir si, à travers ceci, quelque chose de l'informatique a été perçu par les élèves, et si les bénéfices culturels attendus par le séminaire de Sèvres ont été obtenus, nous paraît pour le moins ouverte.     Il faut donc reposer la question : les conclusions de Sèvres, en 1970, sont-elles encore valables en 1980 ? Faut-il introduire maintenant l'informatique comme *discipline en soi* dans l'enseignement secondaire ? Pour éclairer notre réponse à ces questions, nous allons d'abord retracer rapidement l'évolution de l'informatique en tant que discipline scientifique.  **4. La méthode informatique**  ***4.1. Bref rappel historique***     L'histoire nous apprend que l'informatique est issue de la technique. Il y a eu d'abord des ordinateurs (apparus dans les années 1945-1950). Il a fallu en préparer les programmes de travail. Le langage des machines étant particulièrement difficile à maîtriser par l'homme, on a d'abord cherché à en donner des formes plus lisibles, sans agir sur leur structure (langages d'assemblages, vers 1954). Puis on s'est aperçu que l'on pouvait utiliser des formes plus proches de la pensée humaine, et néanmoins compatibles avec la structure des ordinateurs, Fortran est apparu vers 1955.     En moins de 10 ans, on a vu alors naître presque tous les grands langages utilisés aujourd'hui : Cobol, Algol, Lisp, APL, Basic, Snobel... Mais dans le même temps, les ordinateurs voyaient leur structure évoluer très vite. On les dotait de parallélisme, de mécanisme de détection d'erreurs ou d'exceptions, de réserves de mémoire organisées hiérarchiquement. Il a fallu gérer cette complexité, et un effort considérable a été réalisé pour l'écriture de systèmes d'exploitation.     Une véritable course était ainsi engagée entre la technique et sa maîtrise par les utilisateurs. D'autant plus que la technologie évoluait : les machines de 1955 étaient à tubes, celles de 1960 à transistors, celles de 1965 à micromodules, celles de 1970 à circuits intégrés... Cette course a considérablement freiné l'effort de réflexion. Le résultat en a été que la programmation est restée durant toute cette période une activité purement empirique. Son enseignement se limitait à celui d'un langage et ne nécessitait que quelques heures. Corrélativement, tout programmeur devait se forger soi-même ses méthodes de travail. En ce sens, tous les programmeurs étaient autodidactes. Il faudrait en toute rigueur parler au présent. Les programmeurs aujourd'hui en activité, ayant pour la plupart été formés il y a plusieurs années, sont encore des autodidactes. Et si de nouvelles méthodes ont été présentées à certains d'entre eux (j'ai personnellement fait de tels cours à la CII en novembre 1979, à IBM France en mars 1980, et aux programmeurs des sciences humaines en 1980), la routine constitue une pesanteur qui en maintient beaucoup dans les voies qu'ils connaissent, même si elles sont tortueuses...     Tout ceci fait que l'art du programmeur a été longtemps incommunicable. Le programmeur de 1980 n'est pas mieux armé que celui de 1960, s'il n'a pas reçu une formation solide entre temps. Le résultat de l'absence de formation à la programmation, c'était un rapport de performance de 1 à 20 entre des personnes ayant subi le même enseignement,     C'est vers 1965 qu'a commencé à émerger l'idée que l'informatique était plus qu'un phénomène technique, et qu'elle reposait sur une forme de pensée originale qui lui appartenait en propre. Mais c'était alors une intuition plus qu'une évidence tirée des faits.     C'est en 1969 qu'est posée la première pierre de l'édifice qui devait amener la programmation au stade de discipline scientifique. Edsger Dijkstra publie ses notes de « programmation structurée ». Il souligne qu'essayer un programme peut servir à montrer qu'il contient des erreurs, jamais qu'il est juste. On met en évidence l'état lamentable de l'art en matière de programmation (colloque de Montery : *the High cost of software*, septembre 1973). Ce colloque dénonce en particulier la méthode empirique en programmation, consistant à écrire un programme n'importe comment, pour le mettre au point ensuite au terminal (la méthode des clubs informatiques}. Les statistiques de l'US Air Force, données à ce colloque pour l'année 1972, et portant sur une dépense de plus d'un milliard de dollars, donne un coût d'écriture de 75 dollars par instruction, pour un coût de mise au point de 4 000 dollars par instruction. Le département de la défense des États-Unis annonce un coût des erreurs de programmation dans ses services de 1,5 milliards de dollars par an...     Seule la constitution d'une véritable discipline scientifique de la programmation, avec des méthodes originales et rigoureuses, paraît capable de résoudre la difficulté.     Les notes de programmation structurée de Dijkstra en donnent les prémisses. Les travaux de Floyd sur la preuve de programmes apportent la clef du raisonnement informatique en matière de création de programmes. Les livres d'enseignement se transforment. Alors que jusqu'en 1970, les publications en programmation étaient des cours de langages : Fortran, Cobol, APL... On voit enfin paraître des livres de méthodologies, présentant les modes de raisonnement et d'analyse en programmation. Citons entre autres :  *Structured programming* (O. J. Dahl, E. Dijkstra, C. A. R. Hoare) ; *Concise survey of computer Methods* (Peter Naur} ; *Principes of program design* {Jackson) ; *Algorithms + Data Structure = Programs* (N. Wirth) ; *A discipline of programming* (E. Dijkstra) ; *Structured Programming : Theory and Practice* {R. C. Linger, Harlan, D. Mills).     Plusieurs ouvrages étaient traduits en Français :  *Programmation systématique* {N. Wirth, traduit par O. Lecarme) ; *Proverbes de programmation* {H. Ledgard, traduit par J. Arsac).     D'autres livres originaux paraissaient en France :  *Mes premières constructions de programmes* (Gerbier, Springer Verlag, 1977) ;  *La construction de programmes structurés* {J. Arsac, Dunod 1977} ; *Méthodes de programmation* (C. Baudoin, B. Meyer, Eyrolles, 1978).     Vues d'universitaires dira-t-on ! Le succès de la programmation structurée est essentiellement dû à la réussite du système du New York Times dans des conditions encore jamais atteintes {700 000 instructions, et pas plus de 8 jours de mise au point). Le succès en France de la méthode Warnier dans l'informatique de gestion (analogue à la méthode Jackson dont le succès en Angleterre est non moins grand} est un autre témoignage de la valeur de ces méthodes. Disons qu'elles n'ont pas encore atteint le grand public des programmeurs. On estime à 10 ans en moyenne le temps nécessaire pour qu'une invention passe dans les faits. Sachant que les idées nouvelles en matière de programmation ont débouché sur des résultats concrets enseignables en 1977, il ne faut guère attendre d'effet de masse avant 1985.     Reste qui si l'on attend sans rien faire, ils ne seront jamais atteints. Un effort considérable de renouvellement de la pédagogie a été entrepris dans les universités, maintenant presque toutes reconverties à un enseignement sérieux de la programmation (mais non pas toutes : il existe encore des universités ou Écoles où l'on enseigne les organigrammes, pourtant complètement abandonnés dans la pédagogie moderne de la programmation}.     Au terme de ces années d'intense activité de recherche scientifique, pour la plus grande part conduite en Europe, et où la France a joué un rôle considérable, les intuitions de 1970 sur la science informatique et les valeurs culturelles qu'elle développe, se trouvent ainsi justifiées dans les faits.  ***4.2. La méthode informatique***     L'informatique traite l'information, conçue comme le support formel des connaissances. Ce que l'informatique manipule, ce ne sont pas des connaissances, mais seulement des textes formés de caractères et auxquels l'homme peut éventuellement donner un sens. Cette disjonction entre le contenant (l'information) et le contenu (la connaissance) n'est pas nouvelle. Mais elle devient opérationnelle avec l'informatique.     L'homme n'est guère intéressé que par le traitement de connaissances. Une application informatique commence donc par « coder » les connaissances données en informations données, ce qui introduit souvent un certain appauvrissement (comme le montrent bien les sondages ou enquêtes sociologiques : voir la communication de M. Baro à Marseille). Les informations données sont alors traitées comme suites de caractères, par des méthodes syntaxiques et donc automatisables. Pour que ceci puisse être fait sur un ordinateur, il faut rédiger la description de la méthode de traitement (algorithme) dans un langage interprétable par l'ordinateur. Ceci ne veut pas dire qu'il y a deux étapes distinctes : - création de l'algorithme ; - rédaction dans un langage approprié.     S'il y a des cas où il en est ainsi, notamment dans les applications de calcul numérique, le plus souvent, on rédige directement la méthode de traitement dans le langage de programmation ; c'est d'autant plus nécessaire que l'emploi des ordinateurs impose des contraintes qui influent sur le choix de l'algorithme, et que l'on ne peut toujours séparer complètement les deux activités.     Le résultat du traitement est un texte, auquel l'homme donne un sens (interprétation des résultats). Si la codification des données s'est accompagnée d'un appauvrissement, comme on ne peut faire sortir le plus du moins, l'interprétation des résultats demandera de grandes précautions. Les controverses autour des sondages d'opinion en sont un témoignage éclatant. Tout enseignement de l'informatique devrait mettre ce phénomène clairement en évidence sur un exemple.     La programmation elle-même rentre dans ce schéma de séparation de la forme et du sens. L'homme rédige un programme en fonction de la signification des instructions (sémantique du langage). La machine l'interprète à partir de sa seule forme (syntaxe). Il faut donc garantir que ce qui est écrit a bien la signification que l'on avait en tête. Pendant longtemps, la seule façon de faire a consisté à essayer le programme sur des données tests, en espérant que s'il donne dans ce cas de bons résultats, alors il sera toujours correct.     Deux points importants sont à l'origine du renouveau. On a perçu le double rôle du programme. Si l'ordinateur n'en considère que la syntaxe, l'homme l'écrit et le lit par sa sémantique. Un programme est d'abord écrit par un homme et pour être lu par un homme. Il suffit qu'il soit rédigé dans un langage de programmation pour que l'ordinateur l'accepte. Il faut donc soigner la lisibilité du programme. Ceci impose que l'on respecte un certain nombre de règles de stylistique, largement indépendantes du langage utilisé. Cette stylistique a bien sûr son originalité par rapport à ce qu'enseignent les professeurs de lettres, mais elle rejoint les règles profondes de l'expression écrite.     On a aussi perçu la nature véritable d'un programme. Toute instruction qui le compose décrit une action qui modifie des variables et change donc la situation qui existait avant son exécution. Un programme décrit ainsi la suite de transformations qui fait évoluer la situation depuis l'état initial (où les variables ont les valeurs des données) jusqu'à l'état final (où elles ont les valeurs des résultats cherchés). Un programme est difficile à lire parce que cette suite de transformations ne parle pas à l'esprit en elle-même ; on a trop vite perdu le fil des situations. C'est un peu comme avec ces manipulateurs de cartes qui, partant d'une configuration connue de 3 cartes, font des permutations entre elles : au bout d'un très petit nombre de permutations, on ne sait plus où est une des 3 cartes nommément désignée. Pour rendre un programme lisible, il faut expliciter, de place en place la situation que l'on suppose atteinte.     Mais c'est plus qu'une aide à la lecture. C'est une aide à l'écriture. Au lieu d'aligner des instructions en s'en remettant aux essais pour savoir ce qui se passe, on suit l'évolution des situations, mettant à jour la situation au fur et à mesure de la rédaction des instructions, en tenant compte de leur signification. On sait ainsi ce que fait le programme, et que partant de la situation initiale, il produira la situation finale voulue. Ceci a valeur sinon de preuve formelle, du moins de validation sérieuse. Comme on connaît la situation réalisée à chaque pas, et le but cherché, on a un guide pour l'invention des transformations à faire. Il en résulte que bien des algorithmes s'en déduisent de façon assez naturelle. Il n'est plus question dès lors de séparer la création de l'algorithme et l'écriture du programme : les deux activités marchent de pair, la nature opérationnelle du programme fournissant un cadre de pensée utile à l'invention algorithmique. Très souvent, un résultat ne peut être obtenu directement. Il faut s'en rapprocher progressivement, par un chemin en spirale où l'on retrouve périodiquement la même situation générale, mais chaque fois un peu plus près de la situation finale. C'est le mécanisme de l'itération qui est une des grandes structures informatiques, et qui se rattache par cette méthode de raisonnement à la récurrence mathématique.     Il apparaît ainsi que l'enseignement de l'informatique est effectivement un enseignement carrefour. Par les problèmes de syntaxe et sémantique, par les nécessités de la stylistique, il fait un lien avec les enseignements littéraires. Par sa méthode de raisonnement pour la construction et la validation d'un programme, par la récurrence, il rejoint l'esprit de la démonstration mathématique, mais dans un univers beaucoup plus concret.     On pourrait en déduire qu'il vaudrait mieux ne pas enseigner l'informatique en tant que telle, mais laisser le professeur de lettres parler de sémantique et de stylistique, pendant que le professeur de mathématiques parlerait du raisonnement en programmation. Mais on imagine mal comment pourrait se faire un tel découpage, et la façon de coordonner les enseignements qui permettrait à l'élève de voir la profonde unité et cohérence de ces différentes facettes. Ou bien chacun des deux professeurs montrera comment écrire des programmes avec une duplication de l'enseignement, ou bien chacun d'eux sera continuellement obligé de renvoyer à l'autre, le professeur de mathématiques s'abstenant de toute remarque de stylistique lors de la rédaction de programmes, tandis que le professeur de lettres ne pourrait expliquer la signification des programmes dont il étudie la stylistique, pour ne pas redire ce que dit le professeur de mathématiques. On ne peut ainsi couper un enseignement en rondelles présentées séparément.     Cette façon nouvelle de voir la programmation a complètement fait disparaître les organigrammes. C'était des dessins illustrant l'enchaînement d'actions, d'ailleurs plus ou moins bien précisées : seul l'enchaînement était clairement explicité. Or les actions sont secondes. Ce sont les situations qui priment pour la création et la compréhension du programme. L'emploi des organigrammes a fait place à programmation descendante, consistant à faire d'abord un plan du programme, puis à en détailler les différents paragraphes par une méthode comparable : s'ils sont très simples, on les rédige directement sinon, ils correspondent à un nouveau problème, plus simple que le problème initial, et que l'on résout à son tour en l'analysant, puis en faisant un plan, que l'on détaille ensuite, jusqu'à avoir rédigé toutes les parties du programme.     Ces méthodes pénètrent lentement dans le milieu des programmeurs professionnels. C'est ainsi que dans la compagnie IBM France, on parle de pseudo-code au lieu de programmation descendante : l'idée est la même, on sépare le problème en sous-problèmes que l'on traite l'un après l'autre. Les considérations de stylistique ont joué un rôle prépondérant dans la création du langage ADA par la CII. Il est d'ailleurs intéressant de noter que Henry Ledgard, dont les livres de stylistique (*Proverbes de programmation*, 7 éditions différentes) tiennent le record de vente en informatique, est coauteur du langage avec Jean Ichbiah.  **5. Objectifs de l'enseignement**     II ne s'agit pas de former des programmeurs, ni d'ennuyer les élèves avec les détails fastidieux et souvent ridicules d'un langage de programmation : que se passe-t-il si dans un format F il y a plus de caractères à gauche au point décimal dans le nombre que ce qui a été prévu dans le format ? etc. Ceci, c'est l'affaire de professionnels. En ce sens, LSE est beaucoup trop complexe, et l'on ne devrait en présenter aux élèves qu'un sous-ensemble soigneusement choisi pour éviter de nombreux pièges. Si l'on reprend l'idée que la culture, c'est ce qui reste quand on a tout oublié, alors le langage de programmation ne fait pas partie de la culture.     L'important, c'est que l'élève soit capable d'analyser une situation concrète, d'en dégager les tenants et les aboutissants, de dégager la situation initiale et la situation finale, et d'inventer des situations intermédiaires telles qu'on puisse trouver une façon de les enchaîner toutes par des suites de transformations, Il doit pouvoir montrer que ceci résout bien le problème posé.     Il n'est pas question de faire ceci de façon abstraite. On n'attend pas de l'élève qu'il ait acquis des connaissances et sache réciter des théorèmes ou des règles de grammaire. On attend qu'il ait acquis des compétences : en face d'une situation concrète, il sait l'analyser, dégager le problème à résoudre, et, dans des cas pas trop compliqués, imaginer une façon de la résoudre en justifiant ses choix.     Pour ne pas rester dans le vague, voici, à titre purement indicatif, des exemples de situations à étudier.  **1.**Lorsque l'on rédige un chèque, il faut écrire en lettres le montant du chèque. Écrire le programme qui réalise ceci pour un montant qui est un nombre entier de francs supérieur à 1 et inférieur à 1 000.     La difficulté ici vient du grand nombre de règles d'orthographe mises en jeu. Il faudra d'abord que l'élève les retrouve et les réunisse, puis qu'il les organise, par exemple en traitant d'abord la partie des centaines, puis en voyant comment s'organise la partie dizaine-unité.  **2.**Des futurs parents cherchent un prénom pour leur enfant. Mais comme ils ne savent si ce sera un garçon ou une fille, ils veulent un prénom qui puisse convenir aux deux, tel Claude, Camille... Ils ont une encyclopédie qui donne une liste de 100 prénoms masculins et de 100 prénoms féminins.     Cet exemple est fort intéressant parce qu'il montre bien la différence de point de vue avec les mathématiques. Dès l'école primaire, on apprend aux élèves que chacune des listes est un ensemble, et que le résultat cherché est l'intersection des deux ensembles. De sorte que, du point de vue du mathématicien, ceci n'est pas un problème.     On peut imaginer, pour construire cette intersection, de prendre l'un après l'autre les prénoms d'une liste, et de regarder s'ils figurent dans l'autre. Cela fera 10 000 comparaisons.     On peut aussi supposer que les listes sont ordonnées par ordre alphabétique (et au besoin les ordonner). Si le premier élément d'une liste est égal au premier de l'autre, il est dans l'intersection. Sinon, le plus petit des deux dans l'ordre alphabétique ne peut se trouver dans l'autre liste, on peut donc le retirer comme n'appartenant pas à l'intersection. Ceci fera au plus 200 comparaisons.     Là encore, l'exemple est intéressant. Il n'est nullement nécessaire que l'élève invente la deuxième solution. La première est parfaitement valide, et exécutable sur ordinateur en un temps raisonnable. L'élève imaginatif trouvera la seconde solution, et ce sera alors une occasion de parler du problème des performances des ordinateurs et de la complexité des programmes.     Notons enfin que ce problème est un exemple de la classe générale des problèmes de gestion mettant en jeu deux fichiers (fusion, mise à jour...).  **3.**On connaît le jeu dit « master mind ». Le tuteur choisit une combinaison de 4 pions extraits d'un ensemble de pions de 6 couleurs différentes. Le joueur propose une combinaison de 4 pions. Le tuteur répond en disant combien de pions de cette dernière ont la même couleur que le pion de même rang de la combinaison à découvrir, et combien ont la même couleur sans être *à* la même place. Il faut faire tenir à l'ordinateur le rôle de tuteur.     Le problème n'exige que beaucoup de rigueur dans l'organisation si l'on ne veut pas risquer de compter deux fois un pion. On se demande si l'on peut vraiment parler ici d'algorithme, tant les opérations à faire sont simples. Mais l'erreur est vite faite si l'on n'a pas de méthode.     Par ailleurs, ce programme transforme l'ordinateur en partenaire dans un jeu dont la valeur formatrice est évidente. Le joueur ne peut obtenir une bonne stratégie sans démontrer à chaque coup un petit théorème. C'est un excellent jeu d'entraînement au raisonnement.     Comme on le voit, aucun de ces exemples n'a trait au calcul numérique. Il serait peut-être bon de ne pas tomber dans l'excès. Mais s'il faut présenter des exemples de calcul, il vaut sans doute mieux que ce soit fait par le professeur de mathématiques. Car, à partir du moment où un tel enseignement existe, il est clair qu'il est à la disposition de tous les professeurs de toutes les disciplines. On rejoint alors les idées émises par Wladimir Mercouroff et Jacques Hebenstreit. Parce que les élèves ont acquis un minimum de compétence alors le professeur de musique pourra parler d'algorithmes. Le professeur de lettres pourra relier son enseignement de grammaire, ou son souci d'orthographe, à ce qui se passe avec les langages de programmation, et la liaison entre syntaxe et sémantique. Le professeur de physique ou de géographie pourra parler de simulation. L'élève sera en position de comprendre. L'informatique joue bien son rôle de carrefour, la pluridisciplinarité peut intervenir. Nous disons seulement que la méthode informatique est suffisamment importante et originale pour ne pouvoir être présentée à la sauvette, comme une parenthèse dans le cours de lettres ou de géographie (au détriment d'ailleurs de ces disciplines. Le professeur de biologie ne risque-t-il pas de se dire, ou de se faire dire, qu'il ferait mieux d'enseigner la biologie ?)     Il est certes beaucoup trop tôt pour parler du contenu de l'enseignement. Mais on peut déjà prévoir qu'il devrait comporter une présentation générale de l'informatique et de l'information, et une brève description de la structure des ordinateurs.     Il devrait être éminemment concret. Aucune théorie. Le professeur présente des situations concrètes, les analyses avec les élèves, en dégage les problèmes à résoudre, puis entreprend avec eux la construction du programme, leur faisant expliciter les situations à atteindre, la façon de passer de l'une à l'autre. L'élève pourra ensuite passer lui-même le programme sur l'ordinateur, le faire exécuter, étudier son comportement. Après quelques exercices en commun de ce type, l'élève doit alors pouvoir aborder ses propres problèmes (encore que l'expérience montre en général qu'il ne mesure pas la difficulté de ce qu'il entreprend) ou, de préférence, des problèmes que propose le professeur, problèmes pris dans la sphère d'intérêt de la classe. Les problèmes de jeu devraient avoir une place importante : ils apprennent à expliciter des stratégies, ce qui est fort important en informatique.     C'est à l'occasion de l'écriture des programmes que l'on présentera la stylistique, C'est encore à l'occasion du passage de programmes sur ordinateur que l'on pourra parler de performances en temps d'exécution et encombrement en mémoire, et donner quelques notions de complexité.     La question de structuration des données est trop complexe pour être abordée dans ce cadre d'initiation. Les organigrammes devraient être totalement proscrits.     L'enseignement pourra alors valablement parler des possibilités et des limites de l'informatique, présenter les grandes applications, dire ce qui se cache derrière des mots comme robotique, bureautique, fichiers, banques de données, télématique...     C'est à l'enseignement supérieur qu'il appartient de former, ici comme dans toutes les disciplines, les maîtres de l'enseignement du second degré. C'est d'autant plus nécessaire que tout ceci repose sur un effort de recherche récent. Seuls les établissements d'enseignement supérieur ayant déjà une expérience pédagogique en programmation ont compétence pour une telle formation.     Il ne faut pas se cacher la difficulté de la tâche qui attend les professeurs de lycée. Il faut inventer une pédagogie, choisir les bons exemples, forger un vocabulaire simple mais précis, allier la rigueur à la clarté, proposer des méthodes de travail sans nuire à la créativité... Mais l'expérience montre que les professeurs de lycée en sont parfaitement capables.     Ceci veut dire que cet enseignement devrait d'abord être réalisé dans le cadre d'expérimentation pédagogique, avant d'être étendu systématiquement dans des cadres qu'il ne nous appartient pas de définir. Qui sera touché par cet enseignement ? Y aura-t-il un enseignement de base et des options de perfectionnement ? L'enseignement d'initiation sera-t-il lui-même une option ? Toutes ces questions sont laissées ouvertes ici. De même, nous pensons que les professeurs de lycée qui ont participé à l'expérience des 58 lycées sont bien placés pour devenir des enseignants d'informatique, après un complément de formation qui pourrait ne pas être lourd. Mais c'est encore une question qui déborde de ce rapport et de notre compétence.  **6. Conclusion**  - L'informatique devrait entrer dans l'enseignement général comme une discipline à part entière ;  - C'est souhaitable pour trois raisons :   \* le citoyen doit savoir ce qu'est l'informatique qui va modifier si profondément la société dans laquelle il vit,   \* il faut favoriser l'éclosion de vocations chez les jeunes,   \* l'informatique s'appuie sur des méthodes de pensée originales dont l'apport est enrichissant et la valeur culturelle certaine ;  - C'est possible en raison des progrès faits dans la connaissance et la pédagogie de la programmation depuis dix ans ;  - Cet enseignement ne peut être délivré que par des professeurs ayant compétence pour le faire, grâce à une formation solide donnée par des établissements d'enseignement supérieur ayant suivi l'évolution des idées dans la dernière décennie ;  - Ces conclusions prolongent les conclusions du colloque de Sèvres (1970) en les renforçant par les apports récents de la science informatique ;  - Elles ne contredisent pas l'expérience d'utilisation en pédagogie des ressources informatiques, entreprise depuis dix ans. Elles lui donnent au contraire un second souffle en permettant qu'elle s'adresse désormais à des élèves par ailleurs un peu au fait de ce qu'est l'informatique, et en amenant au lycée des professeurs compétents en informatique pouvant aider leurs collègues des autres disciplines ;  - L'enseignement de l'informatique devra être concret, opérationnel, organisateur. Il faudra favoriser au maximum son articulation avec les autres disciplines.  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  *Texte paru en annexe 1 de* L'éducation et l'informatisation de la société. *Rapport au Président de la République sous la direction de Jean-Claude Simon, La Documentation française, 1981, pages 152 à 165.*  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Association EPI |

|  |  |  |
| --- | --- | --- |
| [ccueil](http://www.epi.asso.fr/accueil.htm) | [Informatique et TIC](http://www.epi.asso.fr/revue/editic.htm#h80simon-arsac) | [Historique](http://www.epi.asso.fr/revue/histosom.htm#h80simon-arsac) |