L'enjeu déclaratif

Jean-Marie Ball

L'ENJEU DÉCLARATIF
(ou de l'influence de l'enseignement et de la pratique de la programmation sur la formation intellectuelle)

Jean-Marie BALL

La programmation revient toujours, en fin de compte, à essayer de mettre au pas tout ce qui s'agite à l'intérieur d'une machine pour en obtenir ce que l'on veut. Concrètement cela se traduit aussi par l'élaboration d'un discours dont les imprimantes nous renvoient, quand nous le désirons, les généreuses périodes.

Lutte avec la machine ou lutte avec un discours, le programmeur est-il un meccano de l'abstrait qui remplit un tableau comme on charge une batterie, est-il un poète dont la parole est si musicale et en telle adéquation avec l'ordre du monde, que lorsqu'elle retentit les pierres se meuvent d'elles même pour construire les murailles de Thèbes ?

Même si ces deux attitudes sont efficaces et si l'ordre du monde n'est ici que celui de la machine, il n'est pas sans intérêt de se poser la question, car, suivant la réponse, l'approche intellectuelle des problèmes peut être très différente.

La première attitude correspond à ce qu'on appelle la programmation impérative, où le programme est considéré comme une suite d'ordres ou de commandes par lesquels on indique à la machine comment parvenir au résultat. (Jacques à dit: "Mettez la valeur 47 dans la troisième case du tableau" etc.). Chaque élément du programme correspond à une "action" primitive ou définie par un algorithme.

Cette démarche intellectuelle impérative est l'expression d'une représentation mentale dynamique, elle privilégie la notion d'action, le programme est vécu comme une suite d'événements, comme une aventure au bout de laquelle se profile le résultat convoité. Bien entendu, une méthode rigoureuse, un cahier des charges soigneusement réalisé, peuvent transformer l'aventure en voyage organisé et éviter bien des déboires. Pour mettre au point ou corriger le programme, on le fait "tourner à la main", on essaye d'imaginer ce qui se passe.

La seconde attitude correspond à ce qu'on appelle la programmation déclarative où le programme est considéré comme la phrase définissant son résultat. A chaque élément de programme correspond une définition dont chacun des termes est supposé lui-même défini. Il ne s'agit plus de raconter comment on parvient au résultat mais ce qu'est le résultat.

La démarche intellectuelle déclarative est l'expression d'une représentation mentale statique, elle privilégie la notion d'état, le programme est vécu comme un discours dont l'ambition est de donner une définition exhaustive des termes qu'il produit. Le désir d'exhaustivité induit cet examen casuistique de tous les états possibles qui caractérise la démarche déclarative. Pour mettre au point ou corriger le programme, on s'assure de la correction (et notamment de l'exhaustivité) de chacune des définitions qui le compose.

Même si ces deux façons de concevoir la programmation ont chacune des affinités particulières avec certains langages (LSE, BASIC, PASCAL, sont dits "langages impératifs"; LOGO, LISP et surtout PROLOG, "langages déclaratifs (1)", il n'en reste pas moins qu'avant d'être une affaire de langage, le fait d'être "impératif" ou "déclaratif" est une affaire d'état d'esprit du programmeur.

Cependant l'utilisation de tel ou tel langage induira plus naturellement tel ou tel état d'esprit dont il sera difficile de se libérer par la suite. Alors même qu'elle parle anglais, la vache espagnole est encore toute imprégnée de la représentation mentale du monde qu'impliquent ses racines ibériques (ou euzkadiennes). C'est dire qu'au moment où l'enseignement de l'informatique se répand dans le monde scolaire, il importe de s'interroger sur les schémas de pensée que l'on véhicule et sur la qualité de la formation intellectuelle que l'on apporte.

L'impératif et le déclaratif ne se représentent pas le monde de la même façon. En face de la machine on peut-être Pizarre ou Solon, avoir la dynamique du conquérant qui commande ou la statique du législateur qui rédige. Tous deux sont à la source d'un "ordre", mais ils n'en ont pas la même conception.

Quand l'un ordonne, il commande (imperare = commander), l'ordre est une injonction à agir et sa parole est pour lui moins importante que l'effet qu'elle déclenche.

Quand l'autre ordonne, il range, l'ordre est pour lui un cosmos que sa parole a pour mission de définir et de fonder. Il est plaisant de

constater que nos machines à mettre de l'ordre, nos ordinateurs, expriment par leur radical cette ambiguïté fondamentale.

Bien entendu, quelqu'irréductible que puisse apparaître l'opposition entre ces deux conceptions du monde, elles correspondent cependant à deux aspects complémentaires de la réalité. L'ordre du conquérant n'est qu'une agitation éphémère sans le secours d'un législateur avisé, et le législateur reste bien impuissant sans le solide moteur d'inférences que constitue une société policée.

La distinction reste cependant fondamentale et Montesquieu nous renverrait à la vielle opposition "Législatif-Exécutif". On le sait, la séparation des pouvoirs s'impose dés lors qu'une société atteint un certain niveau de complexité et de développement. Une grande nation ne se gouverne pas comme la bande du quartier. Il en va de même en programmation et les Lumières n'y sont pas moins nécessaires.

L'identité conceptuelle d'un programme est toute entière exprimée dans un langage, elle est toute entière statique et déclarative (pour ne pas dire "législative").

L'exécution d'un programme est toute entière exprimée dans un flux de particules au sein de la machine, elle est donc toute entière dynamique.

La démarche impérative, quant à elle, consiste à n'avoir pas toujours une claire conscience de cette distinction capitale, et à s'efforcer de marcher le plus prés possible du bord de la falaise qui existe entre le programme en tant que discours et le programme en tant qu'exécution. Force est bien de demeurer sur la crête puisqu'en bas ces flux et ces reflux sont du domaine exclusif de la machine, mais le regard plonge et l'âme est fascinée par tout cela qui bouge est paraît si vivant.

Cette sorte de sympathie mécaniste avec la machine qui caractérise la démarche impérative est parfois nécessaire quand il semble que ce n'est qu'en "vivant la machine de l'intérieur" qu'on s'en assurera la maîtrise. Les limites d'une telle attitude, et surtout les dangers qu'elle présente pour la formation des esprits n'ont pourtant pas besoin d'être soulignés.

En règle générale, l'approche impérative reste fondamentalement illusoire, en effet le langage, et partant la programmation, étant d'un autre ordre que l'espace et le temps, le mouvement y est indicible. C'est là l'enseignement essentiel des célèbres paradoxes de Zénon d'Elée (la flèche, Achille et la tortue) qui, s'ils n'empêchent assurément pas la

flèche d'aller se ficher dans la cible ni Achille d'arriver bon premier, montrent bien qu'il n'est possible de parler du mouvement que comme d'une succession d'états. Or toute portion d'espace de temps et de mouvement étant infiniment divisible, il y a donc une infinité d'états où le langage à tôt fait de s'abolir.

Faute de pouvoir "dire" le mouvement, on l'évoque au moyen de procédés stylistiques variés, aspect des temps, locutions (pourquoi pas "tant que" ?) et surtout, si l'on tient à créer une illusion aussi vivante que possible, l'atomisation de l'objet du récit en "états" d'autant plus nombreux que l'on veut être plus prés de la réalité vécue.

De tels récits peuvent être captivants comme un bon film d'aventures, il n'en reste pas moins que le mouvement est ailleurs et qu'entre chaque photo figée de la pellicule il n'y a rien.

Le récit, littéraire ou cinématographique, est une chose, la programmation en est une autre, et lorsqu'on voit s'aligner des lignes de BASIC, action par action, éventuellement obligeamment commentées, le film est nettement moins palpitant.

En définitive, il serait souhaitable à tous points de vue, et surtout si l'on souhaite faire de la programmation un moyen de formation des esprits, de donner à l'approche déclarative une place accrue voire même prépondérante. En posant ses définitions, la programmation déclarative construit un langage et par conséquent un savoir. A côté de la sacro-sainte décomposition cartésienne (Analyse descendante) de problèmes déjà spécifiés, elle permet de donner une plus large place à des approches de synthèse ascendante qui offrent quand même de plus vastes espaces à l'expression et à la créativité. Plus qu'une quelconque habileté à la dissection opératoire, elle apporte la maîtrise attachée à la chose justement dite.

QUELLES CONSÉQUENCES PRATIQUES ?

Quelles conséquences pratiques tirer de cela dans la pratique ou l'enseignement de la programmation ?

Essayer, bien sûr, de faire sentir et penser la programmation de façon aussi déclarative que possible (2).

Notion de nom.

La programmation est un discours et l'une des fonctions essentielles de celui-ci est de pouvoir "nommer". C'est en nommant que le

langage maîtrise et organise le monde. A partir du langage machine, les variables globales puis locales, les procédures, fonctions et prédicats racontent à leur manière comment, peu à peu et avec une puissance et une souplesse croissante, cette fonction de nomination envahit les langages informatiques au fur et à mesure qu'ils sont plus évolués. En BASIC notamment, qui ignore les labels, la nomination est particulièrement malcommode. C'est la raison pour laquelle ce langage vous colle à la cervelle et il est si difficile d'y prendre le recul nécessaire pour exprimer sereinement sa pensée. On pourrait néanmoins pratiquer une programmation déclarative en BASIC, à la condition absolue de ne considérer les numéros de lignes exclusivement que comme des noms en dehors de toute considération de rang. Cela induit des lignes longues et une utilisation abondante du GOSUB mais le résultat est structuré, clair et efficace.

En règle générale, et quel que soit le langage utilisé, il est souhaitable que la décision de créer un Nom relève plus de la cohérence sémantique de ce qu'on veut exprimer (ce nom est nécessaire à la clarté du langage) que du caractère répétitif de certains traitements (ce nom correspond à une procédure commode à utiliser), de toutes façons le souci d'un langage clair induit nécessairement la génération d'un langage et de concepts commodes.

Traitement itératif et récursivité.

La programmation impérative exprime les traitements itératifs à travers la notion de "BOUCLE". Un programme d'une certaine importance est bien souvent conçu comme la gestion d'un grand nombre de boucles, imbriquées ou non, qui permettent au programme de "tourner". C'est la raison pour laquelle ces boucles conditionnelles (Tant Que, Faire Tant Que etc.) sont le lieu de prédilection de la projection mentale mécaniste. Cette projection devient vite si prenante qu'elle constitue pour certains la fine pointe de l'art programmatoire, tandis que d'autres, plus sérieux, s'efforcent de la conjurer à grand coups d'assertions et d'invariants de boucles, afin de retrouver un peu de la staticité et sérénité déclarative.

En vérité, le mérite essentiel de la récursivité est de permettre la définition déclarative des traitements itératifs. Elle permet de "casser" les boucles et par suite de démêler les écheveaux conceptuels qu'elles traînent à leur suite. C'est sans doute lors du passage d'une boucle "Tant Que" à une récursivité terminale, justement par ce qu'au fond c'est

exactement la même chose, que l'on sent le mieux la différence essentielle d'état d'esprit entre l'approche impérative et l'approche déclarative.

Abordées de façon déclarative et récursive, certaines difficultés traditionnelles de programmation disparaissent comme par enchantement. On pense aux fameuses "boucles imbriquées" que l'on maîtrisait à grands coups d'indentations successives et de lignes de commentaires pour indiquer les "branchements". Ce genre de réalisations farfelues est sans doute la meilleure illustration de ce que peut donner au pire l'approche impérative-mécaniste de la programmation (3).

L'utilisation judicieuse de "Noms" et de définitions récursives aurait tôt fait de dégonfler le soufflé. Ce n'est pas le problème d'un traitement itératif de savoir si le traitement qu'il itère en itère un autre à son tour. Quand bien même son discours serait-il exact, ne serait-il pas un peu "shaddock" l'instituteur qui voudrait en une seule et même phrase expliquer à ses élèves l'expansion de l'univers, la rotation des galaxies sur elles-mêmes, des planètes autour du soleil, de la terre sur son axe, des électrons autour des noyaux et du manège sur la place du village, sans oublier la précession des équinoxes ? Il en résulterait peut-être une sorte d'émoi poétique devant le spectacle du mouvement universel, assurément pas une compréhension scientifique de chacun des phénomènes.

Structures de contrôle.

Chacun connaît les trois structures de contrôle à partir desquelles tout algorithme exprimable peut-être théoriquement exprimé. Cette assurance, de prime abord roborative, ne tient pas toujours les promesses de maîtrise intellectuelle qu'on avait pu y placer. Les problèmes intéressants sont bien souvent complètement occultés par la mécanique de leur traitement. L'approche déclarative permet dans une large mesure de se libérer de ce souci du contrôle mécanique des traitements, pour se consacrer à la spécification exhaustive des cas significatifs et des définitions qui leurs correspondent.

Cette méthodologie qui constitue le cœur même de l'approche déclarative me semble très formatrice. Si l'on ouvre un dictionnaire pour trouver la définition d'un mot, on découvre, suivant les cas, qu'il en a une ou plusieurs en fonction des contextes dans lesquels il peut-être utilisé. De la même façon, programmer revient ici pour l'élève à poser les définitions qui lui permettront de construire l'excellent dictionnaire dont

il a besoin pour exprimer ce qu'il souhaite. Qu'il apprenne à définir et structurer correctement son expression et le contrôle opératoire lui sera donné gracieusement en plus. Les pierres se mettront en marche.

CONCLUSION.

Il existe sans doute des situations et des classes de problèmes en face desquels nous ne sommes à priori capables de développer que des représentations mécanistes. Dans ces cas là, comme il faut avancer quand même, il n'y a pas d'autre choix que de penser comme on le peut. Si ultérieurement nous devons acquérir une meilleure maîtrise du problème il est probable que cela coïncidera avec la capacité à en donner une définition déclarative.

En dehors de ces cas, et tout spécialement s'il s'agit d'apprendre à exprimer à travers la machine un savoir d'un certain d'un certain niveau d'abstraction, l'approche déclarative est beaucoup plus appropriée.

De l'impératif au déclaratif on retrouve la distance qu'il y a du commentaire sportif à la déclaration des droits de l'homme, du récit à la loi, de l'opération à la fonction, du savoir-faire à la science.

L'orientation de l'enseignement de la programmation vers des bases plus volontiers déclaratives est donc indispensable si l'on attend de cette discipline une véritable formation de l'esprit plutôt que des exercices de virtuosité à résoudre des casse-tête.

La disponibilité de LOGO, de PROLOG et de Générateurs de Systèmes Experts sur le matériel scolaire rendent désormais cette orientation possible.

Jean Marie BALL.
Formateur au Cafip de St Martin BOULOGNE/MER.

NDLR : une première version de cet article a été publiée dans *B.I.I.P.* n° 5 (Bulletin d'Information sur l'Informatique Pédagogique - académie de Lille) - 1er semestre 1987.

NOTES

1-Pour LOGO ou pour LISP, on parle en réalité de langages "Applicatifs". Ils sont en effet très différents d'un langage purement déclaratif comme PROLOG. Ils me semblent cependant faire tous partie de la famille intellectuelle déclarative. En Lisp ou Logo, par

rapport à la fonction appliquante, la fonction appliquée est dans un statut de subordonnée par rapport à une principale, alors que prolog pratique la coordination.

En Prolog on dira: "Je regarde la muraille de Chine, si je regarde un mur et ce mur a plusieurs milliers de kilomètres".

En Lisp ou Logo: "Je regarde la muraille de Chine, si je regarde un mur qui a plusieurs milliers de kilomètres".

Il s'agit bien dans les deux cas de "poser une définition" et c'est ce qui, dans l'esprit de cet article, fonde le caractère déclaratif.

2- En fait on s'aperçoit que les méthodes de programmation sont développées en fonction des langages dont elles ont à juguler les carences. On retrouve dans la méthode tout ce qui manque au langage. Une telle constatation implique de ne pas confondre la rigueur et élégance de la pensée avec les lourdeurs méthodologiques imputables à des langages de bas niveau. De scribe pharaonique il convient peut-être de devenir professeur de langue.

3- Où l'on confond allègrement structuration de la pensée et mise en page du listing.