

Le pseudo-code, une pseudo-bonne idée

Jean-Christophe Filliâtre

1^{er} octobre 2026

Résumé

En informatique, le pseudo-code est un concept répandu, utilisé dans les ouvrages, les articles de recherche ou encore les articles encyclopédiques pour présenter des algorithmes ou des programmes. Si l'idée semble pertinente, par sa volonté d'abstraction et d'universalité, elle souffre en réalité de nombreux défauts, qui la rendent finalement totalement contre-productive. On résume ici les principaux arguments en défaveur du pseudo-code.

En premier lieu, définissons ce que nous entendons par pseudo-code, si tant est que cela soit possible. Il s'agit d'un langage de programmation idéalisé, avec des éléments de syntaxe (mots-clés, crochets, symbole d'affectation, appels de fonctions, etc.) mais aucune implémentation (ni interprète, ni compilateur, ni même d'outil pour vérifier la bonne formation des programmes). En particulier, le pseudo-code n'est pas la même chose qu'un algorithme écrit en langue naturelle. Ainsi, dans *The Design and Analysis of Computer Algorithms* (Aho, Hopcroft et Ullman, 1974), les auteurs introduisent le tri par base comme ceci,

1. Initialize m empty queues, one for each integer in the range 0 to $m - 1$.
2. Scan the sequence $a_1, a_2, \dots, a_n$ from left to right, placing element a_i in the a_i -th queue.
3. Concatenate the queues (the contents of queue $i + 1$ are appended to the end of queue i) to obtain the sorted sequence.

et il n'y a bien sûr rien à redire à cette façon de faire. On parle de pseudo-code quand les programmes commencent à ressembler à de vrais programmes, mais écrits dans un langage imaginaire. C'est là que les ennuis commencent, que nous détaillons ci-dessous.

Le pseudo-code ne passe pas le filtre d'une vérification, même minimale.

Comme il n'a pas d'implémentation (interprète ou compilateur), aucune vérification n'est faite sur le pseudo-code au-delà de la lecture par les auteurs ou leurs pairs. Il n'est pas rare de trouver dans le pseudo-code des erreurs aussi grossières qu'une arité de fonction non respectée. Ainsi, dans la troisième édition d'un ouvrage pourtant réputé (dont on taira le nom par charité), on trouve la définition d'une fonction avec quatre paramètres formels, appelée quelques pages plus loin avec seulement trois paramètres effectifs. Vérifier l'arité des fonctions n'est pourtant pas chose difficile. De même, on n'aura pas vérifié les types des paramètres, la portée des variables, etc. Au-delà de ces précautions élémentaires, on aura encore moins testé les programmes, en l'absence de toute implémentation.

Le pseudo-code n'a pas de sémantique proprement définie. Quand bien même le pseudo-code ne serait pas destiné à être implémenté, il n'en a pas moins besoin d'une sémantique proprement définie. Pour accepter l'algorithme ou le programme, le lecteur doit le comprendre comme l'auteur l'a voulu. Or, malheureusement, le pseudo-code n'est jamais accompagné d'une sémantique suffisamment précise. Si c'était le cas, on serait en train de définir un nouveau langage de programmation ! (Pourquoi pas.) Toujours dans *The Design and Analysis of Computer Algorithms*, les auteurs font le choix du langage de programmation ALGOL pour écrire leurs programmes. D'une façon intéressante, ils se permettent une petite modification du langage (de l'appel par référence plutôt que de l'appel par nom), mais elle est soigneusement définie. (Et par ailleurs, tout le reste du fragment d'ALGOL utilisé est également soigneusement défini.)

Ce n'est pas plus long d'écrire dans un vrai langage. Dans l'immense majorité des cas, écrire un programme dans un langage de programmation existant n'est pas plus long, ni plus compliqué, que de le faire en pseudo-code. Certes, il faut faire un choix d'un langage, et ce n'est pas facile, mais ce sera toujours mieux que le pseudo-code, car les programmes auront été compilés et testés, et compris correctement par le lecteur (voire réutilisés tels quels par le lecteur). Ainsi, Aho, Hopcroft, Ullman utiliseront PASCAL quelques années plus tard pour un autre ouvrage, *Data Structures and Algorithms*. (Et ils trouveront dans PASCAL l'appel par référence qui leur avait manqué dans ALGOL.) Ils se permettent quelques libertés, avec des expressions conditionnelles parfois écrites en anglais, mais les constructions restent toutes celles du langage PASCAL. Pour un exemple plus récent, on peut citer l'ouvrage *Algorithms* (4th ed.) de Sedgwick et Wayne, où tous les programmes présentés sont écrits en Java (et disponibles en ligne par ailleurs).

En programmation, tout est dans les détails. Or, le pseudo-code ne permet pas, la plupart du temps, d'apprécier toutes les subtilités que pose une réelle implémentation, que ce soit des difficultés de mise en œuvre ou des artefacts particulièrement ingénieux pour l'émerveillement du lecteur. On parle là de choix de structures de données, d'éventuels problèmes de débordement de pile, ou encore de débordements arithmétiques. Et puisqu'on mentionne l'importance des détails, notons enfin qu'un nombre significatif de pseudocodes font le choix d'indexer les tableaux à partir de 1, sans doute influencés par la pratique mathématique. Il va sans dire que la traduction vers un langage de programmation s'avère alors plus que périlleuse.