

Préambule : le Codage**Introduction à l'algorithmique**

Qu'est-ce que l'algomachin ?

Faut-il être matheux ?...

L'ADN, les Shadoks et les ordinateurs

Algorithmique et programmation

Avec quelles conventions écrit-on ?

1. Les Variables**2. Lecture et Ecriture****3. Les Tests****4. Encore de la Logique****5. Les Boucles****6. Les Tableaux****7. Techniques Rusées****8. Tableaux Multidimensionnels****9. Fonctions Prédéfinies****10. Fichiers****11. Procédures et Fonctions****12. Notions Complémentaires****Liens****Souvent Posées Questions**

Rappel : ce cours d'algorithmique et de programmation
est enseigné à l'Université Paris 7,
dans la spécialité PISE du Master MECI (ancien DESS AIGES)
par [Christophe Darmangeat](#)

PARTIE 1

INTRODUCTION A L'ALGORITHMIQUE

« Un langage de programmation est une convention pour donner des ordres à un ordinateur. Ce n'est pas censé être obscur, bizarre et plein de pièges subtils. Ca, ce sont les caractéristiques de la magie. » - Dave Small

« C'est illogique, Capitaine » - Mr Spock

L'algorithmique est un terme d'origine arabe, comme algèbre, amiral ou zénith. Ce n'est pas une excuse pour massacrer son orthographe, ou sa prononciation.

Ainsi, l'algo n'est pas « rythmique », à la différence du bon rock'n roll. L'algo n'est pas non plus « l'agglo ».

Alors, ne confondez pas l'algorithmique avec l'agglo rythmique, qui consiste à poser des parpaings en cadence.

1. Qu'est-ce que l'algomachin ?

Avez-vous déjà ouvert un livre de recettes de cuisine ? Avez vous déjà déchiffré un mode d'emploi traduit directement du coréen pour faire fonctionner un magnétoscope ou un répondeur téléphonique réticent ? Si oui, sans le savoir, vous avez déjà exécuté des algorithmes.

Plus fort : avez-vous déjà indiqué un chemin à un touriste égaré ? Avez vous fait chercher un objet à quelqu'un par téléphone ? Ecrit une lettre anonyme stipulant comment procéder à une remise de rançon ? Si oui, vous avez déjà fabriqué – et fait exécuter – des algorithmes.

Comme quoi, l'algorithmique n'est pas un savoir ésotérique réservé à quelques rares initiés touchés par la grâce divine, mais une aptitude partagée par la totalité de l'humanité. Donc, pas d'excuses...

Un algorithme, c'est une suite d'instructions, qui une fois exécutée correctement, conduit à un résultat donné. Si l'algorithme est juste, le résultat est le résultat voulu, et le touriste se retrouve là où il voulait aller. Si l'algorithme est faux, le résultat est, disons, aléatoire, et décidément, cette saloperie de répondeur ne veut rien savoir.

Complétons toutefois cette définition. Après tout, en effet, si l'algorithme, comme on vient de le dire, n'est qu'une suite d'instructions menant celui qui l'exécute à résoudre un problème, pourquoi ne pas donner comme instruction unique : « résous le problème », et laisser l'interlocuteur se débrouiller avec ça ? A ce tarif, n'importe qui serait champion d'algorithmique sans faire aucun effort. Pas de ça Lisette, ce serait trop facile.

Le malheur (ou le bonheur, tout dépend du point de vue) est que justement, si le touriste vous demande son chemin, c'est qu'il ne le connaît pas. Donc, si on n'est pas un goujat intégral, il ne sert à rien de lui dire de le trouver tout seul. De même les modes d'emploi contiennent généralement (mais pas toujours) un peu plus d'informations que « débrouillez vous pour que ça marche ».

Pour fonctionner, **un algorithme doit donc contenir uniquement des instructions compréhensibles par celui qui devra l'exécuter.** C'est d'ailleurs l'un des points délicats pour les rédacteurs de modes d'emploi : les références culturelles, ou lexicales, des utilisateurs, étant variables, un même mode d'emploi peut être très clair pour certains et parfaitement abscons pour d'autres.

En informatique, heureusement, il n'y a pas ce problème : les choses auxquelles on doit donner des instructions sont les ordinateurs, et ceux-ci ont le bon goût d'être tous strictement aussi idiots les uns que les autres.

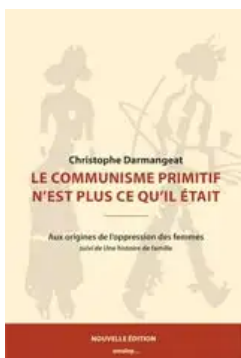
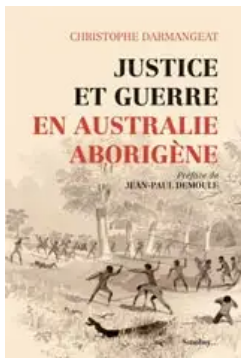


Très loin de l'informatique, pas tout près de l'économie :



Mon blog, [la Hutte des Classes](#)
À propos d'anthropologie sociale, de préhistoire et de marxisme.

Et mes livres...



2. Faut-il être matheux pour être bon en algorithmique ?

Je consacre quelques lignes à cette question, car cette opinion aussi fortement affirmée que faiblement fondée sert régulièrement d'excuse : « moi, de toute façon, je suis mauvais(e) en algo, j'ai jamais rien pigé aux maths ». Faut-il être « bon en maths » pour expliquer correctement son chemin à quelqu'un ? Je vous laisse juger.

La maîtrise de l'algorithmique requiert deux qualités, très complémentaires d'ailleurs :

- il faut avoir une certaine **intuition**, car aucune recette ne permet de savoir a priori quelles instructions permettront d'obtenir le résultat voulu. C'est là, si l'on y tient, qu'intervient la forme « d'intelligence » requise pour l'algorithmique. Alors, c'est certain, il y a des gens qui possèdent au départ davantage cette intuition que les autres. Cependant, et j'insiste sur ce point, les réflexes, cela s'acquiert. Et ce qu'on appelle l'intuition n'est finalement que de l'expérience tellement répétée que le raisonnement, au départ laborieux, finit par devenir « spontané ».
- il faut être **méthodique** et **rigoureux**. En effet, chaque fois qu'on écrit une série d'instructions qu'on croit justes, il faut **systématiquement** se mettre mentalement à la place de la machine qui va les exécuter, armé d'un papier et d'un crayon, afin de vérifier si le résultat obtenu est bien celui que l'on voulait. Cette opération ne requiert pas la moindre once d'intelligence. Mais elle reste néanmoins indispensable, si l'on ne veut pas écrire à l'aveuglette.

Et petit à petit, à force de pratique, vous verrez que vous pourrez faire de plus en plus souvent l'économie de cette dernière étape : l'expérience fera que vous « verrez » le résultat produit par vos instructions, au fur et à mesure que vous les écrirez. Naturellement, cet apprentissage est long, et demande des heures de travail patient. Aussi, dans un premier temps, évitez de sauter les étapes : **la vérification méthodique, pas à pas, de chacun de vos algorithmes représente plus de la moitié du travail à accomplir... et le gage de vos progrès.**



[Retour Haut de Page](#)

3. L'ADN, les Shadoks, et les ordinateurs

Quel rapport me direz-vous ? Eh bien le point commun est : quatre mots de vocabulaire.

L'univers lexical Shadok, c'est bien connu, se limite aux termes « Ga », « Bu », « Zo », et « Meu ». Ce qui leur a tout de même permis de formuler quelques fortes maximes, telles que : « *Mieux vaut pomper et qu'il ne se passe rien, plutôt qu'arrêter de pomper et risquer qu'il se passe quelque chose de pire* » (pour d'autres fortes maximes Shadok, n'hésitez pas à visiter leur [site Internet](#), il y en a toute une collection qui vaut le détour).

L'ADN, qui est en quelque sorte le programme génétique, l'algorithme à la base de construction des êtres vivants, est une chaîne construite à partir de quatre éléments invariables. Ce n'est que le nombre de ces éléments, ainsi que l'ordre dans lequel ils sont arrangés, qui vont déterminer si on obtient une puce ou un éléphant. Et tous autant que nous sommes, splendides réussites de la Nature, avons été construits par un « programme » constitué uniquement de ces quatre briques, ce qui devrait nous inciter à la modestie.

Enfin, les ordinateurs, quels qu'ils soient, ne sont fondamentalement capables de comprendre que quatre catégories d'ordres (en programmation, on n'emploiera pas le terme d'ordre, mais plutôt celui d'**instructions**). Ces quatre familles d'instructions sont :

- l'affectation de variables
- la lecture / écriture
- les tests
- les boucles

Un algorithme informatique se ramène donc toujours au bout du compte à la combinaison de ces quatre petites briques de base. Il peut y en avoir quelques unes, quelques dizaines, et jusqu'à plusieurs centaines de milliers dans certains programmes de gestion. Rassurez-vous, dans le cadre de ce cours, nous n'irons pas jusque là (cependant, la taille d'un algorithme ne conditionne pas en soi sa complexité : de longs algorithmes peuvent être finalement assez simples, et de petits très compliqués).



[Retour Haut de Page](#)

4. Algorithmique et programmation

Pourquoi apprendre l'algorithmique pour apprendre à programmer ? En quoi a-t-on besoin d'un langage spécial, distinct des langages de programmation compréhensibles par les ordinateurs ?

Parce que l'algorithmique exprime les instructions résolvant un problème donné **indépendamment des particularités de tel ou tel langage**. Pour prendre une image, si un programme était une dissertation, l'algorithmique serait le plan, une fois mis de côté la rédaction et l'orthographe. Or, vous savez qu'il vaut mieux faire d'abord le plan et rédiger ensuite que l'inverse...

Apprendre l'algorithmique, c'est apprendre à manier la **structure logique** d'un programme informatique. Cette dimension est présente quelle que soit le langage de programmation ; mais lorsqu'on programme dans un langage (en C, en Visual Basic, etc.) on doit en plus se colleter les problèmes de syntaxe, ou de types d'instructions, propres à ce langage. Apprendre l'algorithmique de manière séparée, c'est donc sérier les difficultés pour mieux les vaincre.

A cela, il faut ajouter que des générations de programmeurs, souvent autodidactes (mais pas toujours, hélas !), ayant directement appris à programmer dans tel ou tel langage, ne font pas mentalement clairement la différence entre ce qui relève de la structure logique générale de toute programmation (les règles fondamentales de l'algorithmique) et ce qui relève du langage particulier qu'ils ont appris. Ces programmeurs, non seulement ont beaucoup plus de mal à passer ensuite à un langage différent, mais encore écrivent bien souvent des programmes qui même s'ils sont justes, restent laborieux. Car on n'ignore pas impunément les règles fondamentales de l'algorithmique... Alors, autant l'apprendre en tant que telle !

Bon, maintenant que j'ai bien fait l'article pour vendre ma marchandise, on va presque pouvoir passer au vif du sujet...



[Retour Haut de Page](#)

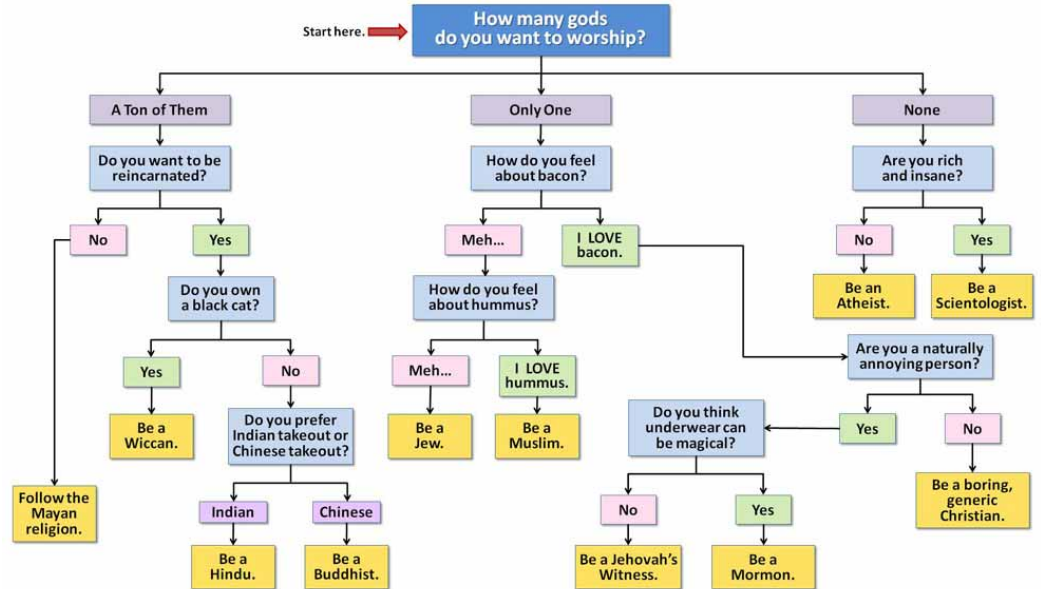
5. Avec quelles conventions écrit-on un algorithme ?

Historiquement, plusieurs types de notations ont représenté des algorithmes.

Il y a eu notamment une représentation graphique, avec des carrés, des losanges, etc. qu'on appelait des **organigrammes**. Aujourd'hui, cette représentation est quasiment abandonnée, pour deux raisons. D'abord, parce que dès que l'algorithme commence à grossir un peu, ce n'est plus pratique du tout du tout. Ensuite parce que cette représentation favorise le glissement vers un certain type de programmation, dite non structurée (nous définirons ce terme plus tard), que l'on tente au contraire d'éviter.

À titre d'exemple, voici la représentation sous forme d'organigramme de l'algorithme qui permet à chacun de choisir la religion qui lui convient :

A Flowchart for Choosing Your Religion



C'est pourquoi on utilise généralement une série de conventions appelée « **pseudo-code** », qui ressemble à un langage de programmation authentique dont on aurait évacué la plupart des problèmes de syntaxe. Ce pseudo-code est susceptible de varier légèrement d'un livre (ou d'un enseignant) à un autre. C'est bien normal : le pseudo-code, encore une fois, est purement conventionnel ; aucune machine n'est censée le reconnaître. Donc, chaque cuisinier peut faire sa sauce à sa guise, avec ses petites épices bien à lui, sans que cela prête à conséquence.

Comme je n'ai pas moins de petites manies que la majorité de mes semblables, le pseudo-code que vous découvrirez dans les pages qui suivent possède quelques spécificités mineures qui ne doivent qu'à mes névroses personnelles.

Rassurez-vous cependant, celles-ci restent dans les limites tout à fait acceptables.

En tout cas, personnellement, je les accepte très bien.



[Retour Haut de Page](#)