

# Comment écrire proprement un algorithme?

Jean-Cédric Chappelier

Version 1.1 – sept. 2018

Ce document donne quelques conseils sur la façon formelle d'écrire un algorithme dans le cours « Information, Calcul et Communication ».

Il se focalise donc sur le style, la *syntaxe*.

Le tout premier conseil est justement de **ne pas** commencer par la syntaxe (« *comment écrire?* ») mais, vraiment, de commencer par le fond/le but (« *quoi écrire?* ») : ne vous bloquez pas sur comment écrire votre algorithme si vous ne savez pas encore clairement ce que vous voulez écrire.

Le premier conseil est donc de réfléchir, faire un/des brouillon(s), schémas, etc.

Une fois au clair sur le « quoi », et seulement à ce moment là, préoccupez-vous de la mise en forme. Commencez pour cela par écrire formellement le problème (en français tout de même) par la description la plus précise possible des entrées fournies à l'algorithme et la sortie obtenue.

Par exemple, pour l'algorithme de recherche d'une des valeurs maximales dans une liste, on écrit :

<b>Valeur maximale</b>
entrée: <i>L une liste non vide de nombres</i>
sortie: <i>la (ou une des) valeur(s) maximale(s) de la liste</i>

Utilisez ensuite les instructions suivantes :

- affectation :  $\leftarrow$   
p.ex. :  $x \leftarrow 3$
- toutes les opérations mathématiques : notation usuelle  
p.ex. :  $x \geq 2$
- désignation d'un élément d'une liste : parenthèses rondes () ou carrées [], au choix  
p.ex. : le  $i$ -ème élément de la liste  $L$  :  $L(i)$  ou  $L[i]$

- les trois structures de contrôle :
  - branchements conditionnels :

```

Si condition
  |
  | instructions
  |
Sinon
  |
  | instructions
  |

```

- boucles conditionnelles :

```

Tant que condition
  |
  | instructions
  |

```

- itérations :

```

Pour tout élément  $x$  de  $L$ 
  |
  | instructions
  |

```

ou

```

Pour  $i$  allant de 1 à  $n$ 
  |
  | instructions
  |

```

Les conventions d'écriture des boucles « Pour tout » incluent :

- que sur les nombres entiers l'incrément est de 1 ; sinon il faut le préciser ;  
p.ex. :

```

Pour  $i$  allant de 1 à  $n$  de 2 en 2
  |
  | instructions
  |

```

autre exemple :

```

Pour  $i$  allant de  $n$  à 1 en descendant
  |
  | instructions
  |

```

- si l'ensemble décrit par la boucle est l'ensemble vide, la boucle ne se déroule pas du tout ; p.ex.

```

Pour  $i$  allant de 1 à  $n$ 
  |
  | instructions
  |

```

ne fera *rien* si  $n$  est inférieur ou égal à 0.

**Remarque :** Pensez à indenter (décaler à droite), et même marquer par une barre verticale, les instructions contrôlées par une structure de contrôle.

- la terminaison de l'algorithme : « **Sortir** : » ;  
p.ex. : **Sortir** :  $x$  ;

Notez que l'instruction « **Sortir** : » met fin à l'algorithme (même s'il y a encore des lignes en dessous).

- si nécessaire (rare dans des algorithmes formels), pour afficher une valeur/expression, utilisez simplement « **Afficher** » ;  
p.ex. : **Afficher**  $x$ .

Sauf mention contraire dans la donnée, vous pouvez également utiliser tout algorithme *vu en cours* (taille, tri, recherche, plus court chemin) en le désignant par un nom suffisamment clair ; par exemple :

- $n \leftarrow \mathbf{taille}(L)$
- $L' \leftarrow \mathbf{trier}(L)$  ou  $L' \leftarrow \mathbf{tri}(L)$

**Note :** au niveau formel, il est préférable de considérer que les algorithmes ne modifient pas leur entrée mais produisent un nouvel objet (comme une fonction mathématique). Par exemple ci-dessus, la liste  $L$  n'est pas modifiée par l'algorithme de tri, mais celui-ci retourne une nouvelle liste (triée).

Voilà pour l'essentiel de nos conseils. Terminons par un exemple complet, un algorithme de recherche d'une des valeurs maximales dans une liste :

<b>Valeur maximale</b>			
entrée: $L$ une liste non vide de nombres sortie: la (ou une des) valeur(s) maximale(s) de la liste			
$n \leftarrow \mathbf{taille}(L)$ $x_{\max} \leftarrow L(1)$ <p><b>Pour</b> <math>i</math> allant de 2 à <math>n</math></p> <table style="border: none; margin-left: 20px;"> <tr> <td style="border-left: 1px solid black; padding-left: 5px;"> <table style="border: none;"> <tr> <td style="border-left: 1px solid black; padding-left: 5px;"> <math display="block">\mathbf{Si} \ L(i) &gt; x_{\max}</math> </td> <td style="padding-left: 5px;"> <math display="block">x_{\max} \leftarrow L(i)</math> </td> </tr> </table> </td> </tr> </table> <p><b>Sortir :</b> <math>x_{\max}</math></p>	<table style="border: none;"> <tr> <td style="border-left: 1px solid black; padding-left: 5px;"> <math display="block">\mathbf{Si} \ L(i) &gt; x_{\max}</math> </td> <td style="padding-left: 5px;"> <math display="block">x_{\max} \leftarrow L(i)</math> </td> </tr> </table>	$\mathbf{Si} \ L(i) > x_{\max}$	$x_{\max} \leftarrow L(i)$
<table style="border: none;"> <tr> <td style="border-left: 1px solid black; padding-left: 5px;"> <math display="block">\mathbf{Si} \ L(i) &gt; x_{\max}</math> </td> <td style="padding-left: 5px;"> <math display="block">x_{\max} \leftarrow L(i)</math> </td> </tr> </table>	$\mathbf{Si} \ L(i) > x_{\max}$	$x_{\max} \leftarrow L(i)$	
$\mathbf{Si} \ L(i) > x_{\max}$	$x_{\max} \leftarrow L(i)$		

Notez que l'algorithme ci-dessus est correct dans tous les cas en raison des conventions :

- la boucle « Pour tout » ne fait rien si  $n$  vaut 1 (et donc, dans ce cas, on retourne finalement  $L(1)$ );
- la description de l'entrée est toujours vraie : ci-dessus la liste  $L$  ne peut (axiomatiquement) pas être vide ; il est donc important de bien préciser les hypothèses de départ. Par exemple l'algorithme suivant n'est pas correct :

<b>Valeur maximale</b>
entrée: $L$ une liste de nombres sortie: la (ou une des) valeur(s) maximale(s) de la liste
$n \leftarrow \mathbf{taille}(L)$ $x_{\max} \leftarrow L(1)$ <p><i>etc.</i></p>

car  $L(1)$  n'est pas défini pour une liste vide (et que l'on n'a pas empêché cette possibilité *a priori*). Il faut donc l'écrire comme donné plus haut, et pas autrement, car il n'y a de toutes façons pas de définition de « la valeur maximale » pour une liste vide.