Glossaire pseudo-code / Javascript

concept	définition	pseudo-code	Javascript
affectation	attribution d'une valeur (par exemple 5) à une variable (par exemple l'entier a)	a ← 5	`a=5;`
affichage dans la console	présentation dans la console d'une valeur entière, d'un flottant, d'une chaîne de caractères ou d'un booléen	afficheConsole("toto")	`console.log("toto");`

algorithme	suite d'instructions pour résoudre un problème, c'est-à-dire fournir un résultat en sortie à partir de variables en entrée. On peut utiliser un algorithme en faisant un "appel d'algorithme" (appel `algo2` dans l'exemple ci-contre) en fournissant les entrées adaptées (`entree1` et `entree2` dans l'exemple) et en récupérant le résultat renvoyé en sortie (dans la variable `var1`, dans l'exemple).	Algorithme algo1
		Entrées : entree1 de type type1, entree2 de type type2.
		Type de sortie : type3
		Variable : var1 de type type3.
		Début
		var1 ← algo2(entree1,entree2)+entree2
		renvoyer var1
		Fin
			`function algo1(entree1, entree2){`
			`  var var1;`
			`  var1 = algo2(entree1,entree2)+entree2;`
			`  return var1;`
			`}`

appel d'algorithme	utilisation d'un algorithme sur d'éventuelles données en entrée	algo1(entree1,entree2)	`algo1(entree1,entree2)`

booléen	valeur vraie ou fausse	Variable : booléen resultat
		resultat ← VRAI
		resultat ← FAUX
			`var resultat;`
			`resultat = true;`
			`resultat = false;`

boucle	structure de programmation qui permet d'exécuter plusieurs fois un suite d'instructions	Tant que . . . faire :
		. . .
		Fin Tant que
			`while(...){`
			`  ...;`
			`}`

chaîne de caractères	ensemble de caractères consécutifs, toujours notée entourée de guillemets droits doubles ou simples	Variable : chaîne de caractères chaine chaine ← "toto"	`var chaine;` `chaine = "toto";`
commentaire	ligne d'un code source qui n'est pas exécutée	//Commentaire	`//Commentaire`
concaténation	opération qui juxtapose deux chaînes de caractères l'une après l'autre pour les réunir en une seule	concatene("toto","1") renvoie "toto1"	`"toto"+"1"` est la chaîne de caractères `"toto1"`
déclaration d'une variable	définition du nom d'une variable	Variables : entier var1 et var2 de type type2.	`var var1; var var2;`
égalité	avoir la même valeur	1=1 ; "toto"="toto"	`1==1; "toto"=='toto'`
entier	en pseudo-code, "nombre entier" ; en Javascript, en fait, nombre à virgule	Variables : entier numero numero ← 42	`var numero = 42;`
entrée	voir algorithme		
flottant	nombre à virgule	Variable : flottant pi pi ← 3.14156	`var pi = 3.14156;`
fonction	voir algorithme		
initialisation	première affectation dans une variable	(voir *affectation*)	(voir *affectation*)
instruction	ordre donné à un ordinateur, élément d'un algorithme	une instruction par ligne	instruction terminée par ";"
longueur	pour un tableau *t*, nombre de cases ; pour une chaîne de caractères *c*, nombre de caractères	longueur(*t*) longueur(*c*)	`t.length;` `c.length;`

modulo	(*a* modulo *b*) est le reste dans la division de *a* par *b*. On peut utiliser le modulo pour tester si un entier *n* est pair.	modulo(*a*,*b*)
		//Test de parité de *n* :
		Si modulo(*n*,2)=0 alors . . .
			`a%b`
			`//Test de parité de n :`
			`if( n%2 == 0 ){ ... }`

renvoyer	donner le résultat d'un algorithme.	renvoyer 42	`return 42;`
sortie	voir algorithme		

tableau	ensemble de variables de même type. La numérotation des cases en Javascript commence à 0.	Variable : tableau d'entiers *t*
		t ← nouveauTableau(2)
		case(*t*,1) ← 18
		case(*t*,2) ← 42
			`var t = [];`
			`t[0] = 18;`
			`t[1] = 42;`
			équivalent à `var t = [18,42];`

test	structure de programmation qui permet d'exécuter une instruction de manière conditionnelle	Si . . . alors :
		. . .
		Sinon :
		. . .
		FinSi
			`if(...){`
			`  ...;`
			`} else {`
			`  ...;`
			`}`

trace	suite des valeurs prises par chaque variable tout au long de l'algorithme	-	-
type	ensemble de valeurs possibles pour une variable	nombre (entier, flottant = nombre à virgule), chaîne de caractères, booléen, mais aussi tableau de ..., date, etc.	`number`, `string`, `boolean`, mais aussi `object`, `bigint`, `function`, `undefined`, `symbol`
valeur	contenu d'une variable, par exemple entier 42, chaîne de caractères "42", ou tableau à deux cases numérotées 4 et 2	42 "42" [4,2]	42 "42" [4,2]
variable	élément ayant un nom fixé, qui permet de stocker une valeur, qui peut changer suite à une affectation	(voir déclaration, initialisation, affectation, type, valeur)	(voir déclaration, initialisation, affectation, type, valeur)

Fiche d'auto-correction

Voici un ensemble de questions que vous devez vous poser après avoir écrit un algorithme en pseudo-code (en Javascript, c'est pareil, il vous suffit de "traduire" les questions ci-dessous en Javascript). Vous devriez toujours répondre oui, sinon cela signifie qu'il y a un oubli ou une erreur à corriger.

Déclaration de l'algorithme

Ai-je déclaré les entrées de l'algorithme ? Ai-je utilisé des noms de variables corrects (commencent par une lettre, pas de caractères spéciaux ni d'espaces) ? Ai-je précisé leur type (s'il s'agit de tableaux, ai-je précisé quel type d'éléments sont stockés dans le tableau ?) ? Correspondent-elles aux entrées indiquées par l'énoncé ?

Ai-je déclaré le type de sortie de l'algorithme ?

Ai-je déclaré les variables de l'algorithme ? Ai-je utilisé des noms de variables corrects (commencent par une lettre, pas de caractères spéciaux ni d'espaces) ? Ai-je précisé leur type ? Ai-je pensé à ne pas redéclarer les variables d'entrée, mais à déclarer l'éventuelle variable de sortie ? Sont-elles utilisées dans l'algorithme, c'est-à-dire entre **Début** et **Fin** ?

Est-ce qu'aucun nom d'algorithme n'apparaît parmi les variables d'entrée et les variables déclarées ?

Instructions de l'algorithme

Pour chaque mot qui apparaît à l'intérieur du corps de l'algorithme, c'est-à-dire entre **Début** et **Fin** :

- S'il est entouré de guillemets, il fait partie d'une chaîne de caractères, ça peut donc être n'importe quel mot.
- Sinon :
  - Si c'est un nom de type en pseudo-code (entier, flottant, booléen, chaîne de caractères, couleur, tableau de ...), ça n'a rien à faire là (les noms de types sont présents uniquement dans les déclarations).
  - Si c'est un élément de langage du pseudo-code (**Si**, **alors**, **Sinon**, **Fin Si**, **Tant que**, **faire**, **Fin Tant que**, **Pour**, **Fin Pour**, **renvoyer**, **case**, **nouveauTableau**, **affiche**) ai-je bien respecté la syntaxe d'utilisation de cet élément du langage (vérifier dans le glossaire) ?
  - Si c'est le nom d'un algorithme (= d'une fonction), il s'agit d'un appel d'algorithme : l'algorithme a-t-il été déclaré auparavant, ou est-il connu d'après l'énoncé ? Ai-je mis entre parenthèses des entrées correctes (bon type, bon nombre d'entrées), séparées par des virgules ? Si l'algorithme renvoie un résultat en sortie, est-ce que je le récupère par une affectation dans une variable (variable du même type que le type de sortie de l'algorithme ?), ou comme entrée d'un autre algorithme (entrée de l'autre algorithme de même type que le type de sortie de l'algorithme ?) ?
  - Si c'est le nom d'une variable, a-t-elle été déclarée parmi les variables ou les variables d'entrée ? De plus :
    - Si elle est à gauche d'une flèche d'affectation, est-ce que son type est le même que ce qui est à droite de la flèche d'affectation ?
    - Sinon, a-t-elle été initialisée précédemment ? S'il s'agit d'une case d'un tableau, est-ce qu'elle existe, et est-elle remplie ? Est-ce que la valeur de cette variable est utilisée d'une manière correcte, soit comme entrée d'un algorithme, soit dans un calcul effectué avec des opérations de base (par exemple l'addition ou la multiplication d'entiers, la concaténation de chaînes de caractères, etc.) ?

Débuguer un code Javascript

Voici les deux principes de base que vous pouvez utiliser pour débugger :

- observer (le comportement du code) pour comprendre (ce qui ne fonctionne pas) : utilisez d'abord la console pour voir s'il y a une erreur, en lisant attentivement le code de l'erreur et en allant voir le numéro de ligne indiqué ; utilisez l'instruction `console.log("message")` pour afficher `message` dans la console afin de vous assurer que le navigateur passe effectivement par cette ligne de code ; si vous utilisez une variable `i`, vous pouvez utiliser `console.log("Valeur de i : "+i)` pour afficher la valeur de `i` au moment voulu ;
- construire un exemple minimal : supprimez des lignes de code qui fonctionnent jusqu'à arriver au code le plus court possible qui ne fonctionne pas, pour identifier "la ligne qui casse tout".