Premiers pas en Algo - D

Objectifs du module

L'objectif de ce module est l'apprentissage des concepts communs à tous les langages informatiques.

Compétences

Approfondissements des notions suivantes:

- boucles
- conditions

Nouveautés:

- tableaux 1D et 2D

Démarche pédagogique

Tous les exercices doivent être réalisés seul(e), en autonomie. Les livrables doivent être validés par le formateur avant de passer à l'exercice suivant.

Exercice D.1 - Les tableaux

Énoncé

Il est souvent nécessaire de stocker un (grand) nombre de valeurs. Par exemple, si on souhaite faire un traitement sur une liste des températures relevées sur plusieurs jours ou afficher la liste de tous les utilisateurs d'un site. La manière la plus simple de faire cela est d'utiliser un tableau.

Écrire une fonction qui, à l'aide d'une boucle, retourne la plus grande valeur du tableau.

Note

Pour créer un tableau, que l'on initialise tout de suite, il faut écrire un code similaire à celui-ci:

```
int[] board = { 7, 2, 9, 10, 1, -4 };
```

Notez, les `[]` après le type `int` qui indiquent à Java que le type est "tableau d'entiers". Il est possible de faire pareil avec les autres types (`float[]` ou `boolean[]`).
Il est ensuite possible d'accéder indépendamment à chaque élément du tableau avec un index que l'on passe entre les crochets. Attention on commence à compter à partir de 0 (donc l'index 1 correspond à la 2ème case, ….)

```
// lit la première valeur du tableau
int anyVar = board[0];

// remplace la 5ème valeur du tableau par 3 (affectation)
myData[4] = 3;

// affiche la taille du tableau
println(myData.length);
```

Livrables

- Une fonction getMax qui prend un tableau en paramètre et retourne la plus grande valeur de celui-ci
- Appeler la fonction getMax avec le tableau `board` ci-dessus.

Ressources

- https://fr.flossmanuals.net/processing/les-listes/
- https://www.w3schools.com/java/java_arrays.asp
- https://processing.org/reference/Array.html
- https://www.tutorialspoint.com/java/java_arrays.htm

Compétences

- Programmer une boucle
- **Programmer avec des tableaux**

Exercice D.1.2 - Somme

Énoncé

Écrire une fonction `getSum` qui retourne la somme de tous les éléments du tableau.

Exercice D.1.3 - Moyenne

Énoncé

Écrire une fonction `getMean` qui retourne la moyenne de tous les éléments du tableau.

Exercice D.2 - Toto, les pieds dans l'tableau

Énoncé

Repartez de la dernière version fonctionnelle de Toto et modifiez le, pour stocker la nature du sol sous la forme d'un tableau de booléens : true indiquant la présence d'une flaque et false un sol sec.

Notes

Pensez à modifier la fonction `isNearWater` ;-).

Vous pouvez maintenant faire avancer Toto sur différents terrains et si votre algorithme est bon, Toto ne devrait jamais avoir les pieds mouillés.

Compétences

- Programmer une condition
- Programmer une boucle
- **Programmer avec des tableaux**

Exercice D.3.1 - Démineur en ligne - Initialisation

Énoncé

Écrire un programme qui initialise un tableau de taille N et place aléatoirement 3 mines dans celui-ci.

Livrables

- Une fonction `initBoard` qui a pour paramètres: la taille de tableau à créer et un nombre de mines à placer dedans. Cette fonction alloue un tableau, à la taille requise, puis l'initialise avec des 0 puis ajoute autant de mines que nécessaire, en tirant aléatoirement la position de celles-ci (une mine étant représentée par la valeur 99). Il n'est pas nécessaire quand on place une mine aléatoirement de vérifier si une mine est déjà présente à cet endroit.
- un affichage du tableau dans la console après son initialisation avec un tableau de taille 10 avec 3 mines

Ressources

- Pour allouer des tableaux : https://miashs-www.u-ga.fr/prevert/Prog/Java/CoursJava/tableaux.html
- Pour choisir un nombre aléatoirement : https://waytolearnx.com/2018/11/comment-generer-facilement-des-nombres-aleatoires-en-java.html

Exercice D.3.2 - Démineur en ligne - Parcours

Énoncé

Pour chacune des cases qui ne contient pas de mine, calculer le nombre de mines dans les cases adjacentes.

Livrables

Un évolution du programme précédent avec :

- Une fonction `isAMine` qui prend en paramètre un index et indique si la case contient une mine
- Une fonction `getMines` qui prend en paramètre un index et retourne le nombre de mines dans les cases à côté.
- un affichage dans la console du tableau où, pour chaque élément du tableau, on affiche un X quand il y a une mine, ou le nombre de mines dans les cases adjacentes s' il s'agit d'une case vide.

Exercice D.4 - Démineur en 2 dimensions

Énoncé

Modifiez le programme précédent pour faire un démineur en deux dimensions.

Notes

Toutes les fonctions sont à adapter mais la logique reste la même, à la différence qu'il faut maintenant deux index (une pour la ligne, une pour la colonne) pour localiser une case du tableau. Il faut donc tirer deux nombres aléatoires pour positionner une mine, pour tester une case, il faut regarder les 8 cases autour, …

Ressources

- https://waytolearnx.com/2020/04/comment-declarer-initialiser-et-afficher-un-tableau-a-deux-dimensions-java.html
- https://miashs-www.u-ga.fr/prevert/Prog/Java/CoursJava/tableaux.html

Exercice bonus - Démineur 2D graphique

Énoncé
Faire une version graphique du démineur 🙌

Livrables
Un programme avec les fonctions suivantes.
Pensez à tester à chaque étape.

- Une fonction qui affiche une case en fonction de ses coordonnées (rappelez-vous la fonction rect)
- Une fonction draw qui affiche toutes les cases (la fonction draw comme la fonction setup est appelée automatiquement par processing)
- Quand on clique sur une case (voir mouseClicked), utilisez les variables existantes mouseX et mouseY pour en déduire la case correspondante.
- Si il y a une mine, on affiche un X et on a perdu, sinon on affiche dans la case le nombre de mines adjacentes (fonction pour afficher du texte)
- Quand toutes les cases, ne contenant pas de mines, ont été découvertes, le joueur a gagné

Ressources
- https://processing.org/reference/mouseClicked_.html
- https://processing.org/reference/mouseX.html et https://processing.org/reference/mouseY.html
- https://processing.org/reference/text_.html