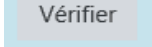



1. I2010 : langage C (TP2)

Avant de résoudre les exercices de cette fiche, faites le petit test CodeRunner « **TP2 Les tableaux - Quiz** » sur Moodle.

Pour chaque question du quiz, il faut d'abord cliquer sur le bouton  avant de cliquer sur « *Page suivante* » afin de vérifier que votre solution compile et passe les différents tests.

Conseil : afin d'avoir une meilleure visibilité des questions, cachez le volet gauche de Moodle grâce à l'icône (en haut à gauche de la fenêtre) : 

Il est également possible d'agrandir la cadre contenant le code à compléter en tirant sur le coin inférieur droit :



Tableau à 1 dimension

1. Ecrire un programme qui permet à l'utilisateur de calculer certaines statistiques sur les notes d'une classe.

Le programme lit des notes X_i au clavier via la fonction `scanf`¹. On supposera qu'il y a maximum 50 étudiants dans une classe. L'utilisateur arrête l'introduction des notes en entrant Ctrl-D.

Le programme calcule :

- la moyenne de la classe $\text{Moy}(X_i)$,
- les écarts à la moyenne $(X_i - \text{Moy}(X_i))$ pour chaque valeur X_i ,
- ainsi que la variance, i.e. la moyenne des écarts au carré : $\text{Moy}((X_i - \text{Moy}(X_i))^2)$.

Essayez de minimiser le nombre de parcours de votre tableau de notes. Au total, deux parcours devraient suffire.

¹ cf. eSyllabus chap. 6 ou le manuel `man scanf`.

2. Un nombre premier est un nombre entier qui n'est divisible que par deux diviseurs : 1 et lui-même.

La recherche des nombres premiers a titillé la sagacité des mathématiciens depuis la nuit des temps. Au III^{ème} siècle avant J-C, en Grèce, le philosophe et mathématicien Eratosthène a imaginé une technique pour retrouver la liste des nombres premiers.

Son procédé, *le crible d'Eratosthène*, consiste, pour K allant de 2 à N, à parcourir la liste des nombres de K à N en supprimant tous les multiples de K. Les nombres qui n'auront pas été supprimés constituent la liste des nombres premiers inférieurs à N.

Il n'est bien entendu pas utile de (re)supprimer les multiples des nombres déjà supprimés. De même, les parcours peuvent s'interrompre dès que K atteint la valeur de la racine carrée de N.²

Par exemple, si le programme doit rechercher les nombres premiers compris entre 2 et 10, les étapes seront les suivantes:

| | | | | | | | | | |
|---------|---|---|---|---|---|---|---|---|----|
| Etape 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| Etape 2 | 2 | 3 | | 5 | | 7 | | 9 | |

où le nombre en bleu correspond à K, càd celui dont on élimine les multiples ; les multiples de K sont indiqués dans les cases rouges et les nombres premiers trouvés sont les nombres restants, dans les cases vertes. Dans cet exemple, la recherche peut s'arrêter dès l'étape 2 étant donné que $4 > \sqrt{10}$, où K=4 et N=10. Les nombres premiers inférieurs à 10 sont donc : 2, 3, 5, 7.

Pour avoir une démonstration visuelle, rendez-vous sur le site :

https://fr.wikipedia.org/wiki/Crible_d%27%C3%89ratosth%C3%A8ne

Il vous est demandé d'écrire un programme qui affiche les nombres premiers compris entre 2 et N grâce à la méthode décrite ci-dessus. Cette méthode ne nécessite ni division, ni multiplication ! La valeur de N doit être strictement inférieure à 100 et être lue grâce à la fonction `scanf`. Veuillez à vérifier que la donnée entrée par l'utilisateur est comprise entre 2 et 100. Si ce n'est pas le cas, demandez à l'utilisateur d'en entrer une nouvelle.

Indices de solution :

utilisez un *tableau de booléens*, où les éléments d'indices 0 et 1 ne sont pas utilisés.

² La racine carrée se calcule via la fonction `sqrt` de la librairie `math.h`. Cette librairie nécessite une option de compilation spécifique : « `-lm` ».

Tableau à deux dimensions

3. Écrire un programme qui réserve une table de m lignes et n colonnes dont les éléments sont des caractères (m et n sont 2 chiffres lus sur *stdin* avec *scanf*). On remplit la table ligne par ligne avec des lettres majuscules, en commençant à la lettre 'A' et en continuant jusqu'à la lettre 'Z', puis en recommençant à 'A' et ainsi de suite jusqu'à ce que la table soit remplie.

Exemple pour $m = 5$ et $n = 7$:

| | | | | | | |
|---|---|---|---|---|---|---|
| A | B | C | D | E | F | G |
| H | I | J | K | L | M | N |
| O | P | Q | R | S | T | U |
| V | W | X | Y | Z | A | B |
| C | D | E | F | G | H | I |

On remplit ensuite une nouvelle table de la même manière mais colonne par colonne.

On vous demande d'afficher les caractères, numéros de lignes et numéros de colonnes des cases ayant reçu la même valeur dans le remplissage par ligne et celui par colonne.

Avec l'exemple précédent, on obtient en sortie :

```
A en ligne 0 et colonne 0
R en ligne 2 et colonne 3
I en ligne 4 et colonne 6
```

4. Modifier le programme précédent pour remplir la seconde table, ligne par ligne, avec des lettres choisies au hasard³.
5. Pour terminer, votre programme doit afficher pour chaque ligne, la fréquence de chaque lettre. Par exemple, avec la table remplie au hasard :

| | | | | | | |
|---|---|---|---|---|---|---|
| I | I | K | R | H | U | Y |
| J | K | H | L | H | V | R |
| I | Q | N | I | A | X | M |
| X | Y | M | G | B | Q | L |
| I | T | X | F | Y | W | A |

3. Pour obtenir un nombre aléatoire compris entre *min* et *max*, vous devez utiliser la formule générique : $\text{min} + (\text{int})(\text{rand}() / (\text{RAND_MAX} + 1.0) * (\text{max} - \text{min} + 1))$ où *rand()* et *RAND_MAX* sont déclarés dans le fichier *stdlib.h*.

Le programme affichera :

| | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
|-----------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Ligne 1 : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 2 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| Ligne 2 : | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| Ligne 3 : | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| Ligne 4 : | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| Ligne 5 : | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 |

Bonus : rand()

Exécutez plusieurs fois votre second programme (exercice 4) en entrant les mêmes données (nombre de lignes et nombre de colonnes) et comparez les résultats. Que remarquez-vous ?

Expliquez le comportement du programme et modifiez-le afin de lever la limitation observée.