1 Instructions pour le TP noté

En temps normal, le sujet précise "Supports de cours et TP personnels autorisés, **internet et emails interdits**".

Les circonstances exceptionnelles de cette année 2020 vous forçant à faire ce TP depuis chez vous, nous ne pouvons matériellement pas vous empêcher d'utiliser internet ou de communiquer entre vous.

Nous comptons sur votre honnêteté et votre sens des responsabilités pour résister à la tentation de fraude.

Néanmoins, pour ce TP noté plus que jamais, toute similitude suspecte entre deux codes, ou toute utilisation de morceaux de codes provenant d'internet sera évidemment sanctionnée.

2 Marche aléatoire

Imaginez un homme se déplaçant sur une grille en 2 dimensions de façon aléatoire. Certains parlent de la "marche de l'homme ivre" puisque ses déplacements sont aléatoires ☺.

Il part de la case de coordonnées $(0, 0)$ et peut effectuer 4 déplacements : vers le haut, vers le bas, vers la gauche ou vers la droite.

Voici un exemple de coordonnées successives après que l'individu a effectué 5 pas :

```
(0,0)
(-1,0)
(0,0)
(0,1)
(0,2)
(0,3)
```

Si on représente cela sur un graphique, on obtient la Figure 1.

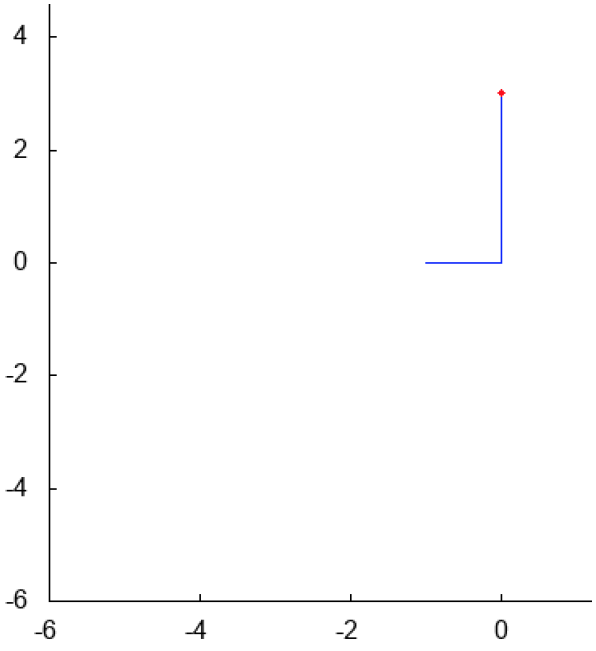

FIGURE 1 –

Remarque : il n'y a **aucune** contrainte sur le déplacement, le marcheur peut revenir sur ses pas [1], comme c'est d'ailleurs le cas sur l'exemple précédent.

La Figure 2 vous donne un autre exemple de marche aléatoire après quelques dizaines de déplacements (le point rouge marque la position finale).

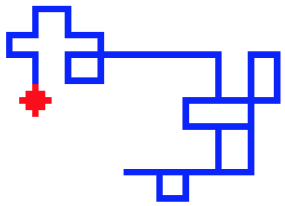

FIGURE 2 –

En mathématiques, il est possible de démontrer que pour une marche de n pas, en moyenne, l'abscisse ou l'ordonnée maximale atteinte se trouvera aux alentours de $\sqrt{n}$.

Néanmoins, si on s'autorise à recommencer l'expérience un grand nombre de fois, il est possible d'obtenir une marche aléatoire où l'individu atteindra un point significativement plus éloigné de l'origine.

Appelons `limite` l'abscisse ou l'ordonnée maximale que notre individu devra atteindre. Il est important de fixer un seuil maximal pour le nombre de pas, appelons-le `max_step`.

En effet, sans ce seuil, si on autorise notre individu à marcher "pour toujours", il atteindra nécessairement `limite` à un moment donné.

Pour vous donner un exemple concret, prenons :

```
limite=50
max_step=400.
```

Cela veut dire que je vais répéter l'expérience de la marche aléatoire jusqu'à ce que l'homme atteigne un point dont l'une des coordonnées est 50 ou -50

1. Il est même tout à fait possible que dès le second pas, le marcheur revienne à l'origine !

(on regarde en fait la valeur absolue). Mais, si après 400 pas, l'individu n'a pas atteint un tel point, je recommence l'expérience.

Il y a alors 2 autres valeurs qui nous intéressent, combien de fois a-t-on dû ré-itérer l'expérience ? Lors de l'expérience qui fut un succès, après combien de pas l'individu a-t-il atteint la limite ?

Pour les valeurs énoncées ci-dessus, une de mes exécutions a donné les résultats suivants :

— Il aura fallu 725 marches aléatoires pour avoir un succès.
— Lors de la marche aléatoire qui fut un succès, c'est-à-dire la 725ème, l'homme a atteint la case $(-50, -17)$ après 367 pas.

Une animation de cette dernière marche aléatoire est disponible dans le dossier du TP noté, il s'agit de `marche_aleatoire.gif`.

Le but de ce TP est de vous faire générer un fichier contenant les coordonnées successives d'une marche aléatoire pendant laquelle l'individu aura atteint une limite de 90 en faisant au maximum 1000 pas. Il sera important de conserver le nombre de marches aléatoires effectuées pour parvenir à ce résultat.

Le rendu de ce projet se limitera à une archive .zip contenant un fichier `random_walk.c` et, si vous avez atteint la Partie 2, un fichier `data.txt` contenant les coordonnées successives de la marche qui aura été un succès.

3 Partie 1 : affichage dans le terminal

Dans le `main` de votre programme, créez un tableau de deux `int`. Celui-ci contiendra les coordonnées de votre marcheur.

Question 1 : Commencez par créer une fonction

```
void deplacement(int *position)
```

qui recevra un pointeur vers le tableau contenant les coordonnées dans votre `main` et qui effectuera **un seul** pas aléatoire. Vous utiliserez pour cela la fonction `rand()` vue en TP.

Question 2 : Faites en sorte que les arguments `limite` et `max_step` soient saisis dans le terminal au lancement du programme.

Question 3 : Votre programme doit alors afficher toutes les coordonnées successives de votre marcheur, tant que celui-ci n'est pas parvenu à un succès. À la fin vous afficherez une phrase bilan du type

```
success in XXX attempts with XXX steps.
```

Astuce : vous pouvez utiliser la fonction `abs()` de la bibliothèque `math.h` pour obtenir la valeur absolue d'un nombre.

Un exemple d'exécution avec les arguments suivants

[`limite` = 3 ; `max_step` = 7]

vous est donné par la Figure 3. Attention, il y a bien eu 12 marches en tout, comme indiqué dans la phrase finale, je n'ai affiché que les dernières.

```
(0,0)
(1,0)
(1,1)
(0,1)
(-1,1)
(-1,0)
(-1,-1)
(-2,-1)
success=0
(0,0)
(0,-1)
(-1,-1)
(-1,0)
(0,0)
(0,-1)
(0,-2)
(1,-2)
success=0
(0,0)
(-1,0)
(0,0)
(0,1)
(0,2)
(0,3)
success=1
success in 12 attempts with 5 steps
```

FIGURE 3 –

4 Partie 2 : écrire les positions dans un fichier

Dans cette partie, nous allons écrire les positions successives de votre marcheur, en partant de $(0, 0)$, dans un fichier.

Attention, à la fin, seule la marche qui aura été un succès sera écrite dans le fichier.

Le fichier contiendra des positions sous la forme suivante

```
0       0
1       0
1       -1
```

C'est-à-dire : abscisse, tabulation, ordonnée, retour à la ligne, abscisse, tabulation, ordonnée, retour à la ligne, etc...

Votre fichier devra être lisible au format texte [2].

Question 1 : En considérant que la valeur absolue de chaque coordonnée peut valoir au maximum `limite`, combien de caractères (octets) peut contenir votre fichier au maximum ? Merci de détaillez votre calcul en commentaire.

Astuce : le nombre de caractères nécessaires pour stocker un nombre `n` *se calcule de la manière suivante en C :* `ceil(log10(n))`*. N'oubliez pas d'importer la bibliothèque* `math.h`*.*

Question 2 : Déclarez un tableau **dynamique** [3] de caractères de la taille que vous avez calculée à la question précédente. Naturellement cette taille va dépendre de `limite` et `max_step`.

La syntaxe de la fonction `sprintf` est la suivante :

```
sprintf(adresse,chaine formatée,...)
```

Les "..." désignent les éléments utilisés par la chaîne formatée, par exemple l'instruction

```
sprintf(p,"n=%d",2)
```

écrira la chaîne de caractères "n=2" dans un tableau de caractères en commençant par la case mémoire d'adresse p.

La valeur de retour de la fonction `sprintf` vous servira par la suite, celle-ci indique le nombre exact de caractères écrits par la fonction dans le tableau, par exemple 3 dans l'exemple ci-dessus.

2. En d'autres termes, contrairement aux fichiers généraux manipulés lors de certains de nos TPs, celui-ci devra être parfaitement lisible par un éditeur de texte.

3. Ceux qui m'ont bien écouté en TP savent parfaitement pourquoi j'insiste sur le mot "dynamique"... En gros, je ne veux pas voir de crochet dans votre code à cet endroit là ☺.

Question 3 : [Question de cours] Pas besoin de coder, merci d'indiquer votre réponse en commentaire dans votre code.

Sachant que la fonction `sprintf` écrit des chaînes de caractères **valides**, et en considérant que la valeur de retour vaut n, que contient la $n+1$-ième case du tableau de caractères ?

Remarque : c'est à cause de ce phénomène qu'il faut toujours déclarer un tableau de taille $n+1$ pour écrire une chaîne de caractères contenant n caractères.

Question 4 : Maintenant que vous êtes à l'aise avec la fonction `sprintf`, écrivez, bout à bout, les positions successives de votre marcheur dans le tableau de caractères déclaré à la question 2.

Naturellement, puisque c'est seulement "à la fin" d'une marche que vous saurez si c'est la bonne ou non, **ce tableau devra être ré-alloué à chaque nouvelle marche.**

Question 5 : Vérifiez avec des petites valeurs que votre chaîne de caractères (celle qui a été créée à la question 4) contient bien ce que vous souhaitez ; vous pouvez alors l'écrire dans un fichier à l'aide la commande suivante

```
fprintf(fichier,"%s",chaine)
```

où `fichier` est un flot, comme vu en TP.

Vous rendrez alors ce fichier dans votre archive, sous le nom `data.txt`.

N'oubliez pas d'indiquer, en commentaire de votre code, le nombre de marches que votre marcheur aura effectuées avant de parvenir à ce succès.

Remarque : c'est un fichier comme celui-ci qui m'a permis de générer la petite animation mentionnée précédemment.