Exercices C – partie 2

Pour tous les exercices il vous est demandé de respecter la structure de code proposée en cours via le fichier structureProgrammeC.c.
Vous veillerez également à assurer la lisibilité de vos codes et pour cela à les commenter, les indenter, à choisir les noms de vos variables et de vos fonctions.
Vous réaliserez également pour chaque programme ou groupes d'exercices un MakeFile associé, permettant de faire le build de l'exécutable ainsi que le clean à posteriori.
Vous soignerez également vos tests.

A. Faire l'exercice B.1 du TD 1.

B. Nombres aléatoires (cf cours pseudo-aléatoire)

1. Librairie

Écrire une série de fonctions, qui génèrent et retournent des nombres aléatoires selon les contraintes suivantes :

- un nombre entier aléatoire selon la plage maximum du générateur aléatoire.
- un nombre entier aléatoire compris entre 0 et une valeur "seuil haut" passée en paramètre.
- un nombre entier aléatoire entier compris entre deux bornes "seuil bas" et une valeur "seuil haut" passées en paramètres.
- un nombre réel aléatoire compris entre 0 et 1.
- un nombre réel aléatoire à deux décimales compris entre deux bornes "seuil bas" et une valeur "seuil haut".

Faire une librairie `util_rand.c` et `util_rand.h` avec ces fonctions.

2. Dés

Écrire un programme affichant un menu proposant de jouer avec un, deux, trois ou quatre dés.
Selon le choix fait, le programme lance les dés tous en même temps. Les dés identiques sont systématiquement relancés.
On calcule alors le cumul des points des différents dés.
L'utilisateur gagne si le total est supérieur aux deux tiers du maximum qu'on peut obtenir avec les dés lancés (avec deux dés cela fait 8 : (12/3)*2). Le programme indique combien il manque pour gagner ou combien il y a en plus.
On utilisera explicitement la librairie écrite au A.1 pour gérer les nombres aléatoires.
Vous manipulerez toutes les variables via des pointeurs…
Vous créerez les fonctions/procédures nécessaires.
Exemples d'interaction :

```
Avec combien de dés voulez vous jouer ?
Tapez 1, 2, 3 ou 4 ? 3
Vous avez choisi 3 dé(s)
Dé 0 : 3
Dé 1 : 4
Dé 2 : 4
Dés identiques relancés...
Dé 0 : 3
Dé 1 : 6
Dé 2 : 1
La somme des 3 dés lancés est de 10, le seuil était de 12
score inférieur de 2 au seuil
Désolé, vous avez perdu !
```

3. Test statistique de rand [Optionnel, pour les curieux]

Écrire une fonction qui tire aléatoirement 100000 fois une valeur comprise entre 0 et 5, compte les occurrences de chaque résultat et en affiche le pourcentage. Testez ensuite cette fonction depuis un main.
On utilisera explicitement la librairie écrite au A.1 pour gérer les nombres aléatoires.

C. Structures, fourmillement

On souhaite implémenter un petit jeu dans lequel une fourmi se déplace à l'écran (La taille maximum de la zone de jeu est de 400 px en largeur et 200 px en hauteur, l'origine de la zone de jeu est en haut à gauche). Pour cela et en préalable on vous demande de proposer une structure fourmi permettant de la représenter. Les caractéristiques d'une fourmi sont les suivantes. Une fourmi possède :

- une position courante donnée par une coordonnée x et y, entiers, qui doit être contenue dans la zone de jeu ;
- une vitesse de déplacement donnée par deux valeurs dx et dy qui représentent respectivement le nombre de pixels en x et en y dont la fourmi se déplace selon l'axe des x et des y ;
- une couleur, représentée par un entier, comprise entre 0 et 255.
- un symbole sous la forme d'un caractère lettre majuscule qui représentera la fourmi à l'écran lors de ses déplacements.

a- Dans un fichier fourmi.h déclarez le type Fourmi permettant de représenter les informations concernant une fourmi ainsi que les déclarations de fonctions suivantes :
   i. Une fonction init qui retourne une fourmi avec tous ses champs initialisés à partir de valeurs aléatoires. Elle ne prend donc aucun paramètre en entrée.
   ii. Une procédure avance qui prend en paramètre un pointeur sur une fourmi et la fait avancer de dx et dy. Cette fonction ne retourne rien.
   iii. Une procédure affiche_p qui prend en paramètre une fourmi et affiche avec printf les valeurs des différents champs de la fourmi.
   iv. [Pour ceux qui sont à l'aise et en avance…] Une procédure affiche_c qui prend en paramètre une fourmi et affiche sur la console la fourmi à sa place.

b- Dans un fichier fourmi.c donnez du code aux fonctions/procédures précédentes.

c- [Pour ceux qui sont à l'aise et en avance qui ont fini **tout le TD**…] Récupérez le fichier `jeuModeConsole_0.c` sur l'ENT et essayez de vous en inspirer pour, en plus, faire bouger en mode graphique console votre fourmi…