Sommaire

TP Fonctions

Fonction moyenne

Question 1 : Ecrire une fonction moyenne qui prend en argument deux variables flottantes x et y, et qui retourne la moyenne de x et y.

Question 2 : Tester votre fonction (il faudra l'appeler !) en affichant dans le main la moyenne entre 10 et 13, puis la moyenne entre 12345.75 et -8361.125.

♪ Le décompte final ♫

1. Écrire une fonction affiche_decompte qui prend en paramètre un entier n et qui affiche tous les nombres de n à 0, en décroissant de 1 en 1, sur une même ligne sans espaces. On effectuera à la fin de cette ligne un retour à la ligne (caractère '\n'). La fonction affiche_decompte ne renvoie rien.

Exemple

L'appel affiche_decompte(7) devra afficher

```
76543210
```

2. Écrire une fonction affiche_triangle qui prend en paramètre un entier n et qui affiche un "triangle rectangle d'entiers", dans lequel au coin en haut à gauche se trouve l'entier n, puis on complètera le triangle en baissant la valeur de 1 chaque fois qu'on descend ou qu'on va à droite. La fonction affiche_triangle ne renvoie rien. (Si ce n'est pas clair, voir l'exemple plus bas)

   On utilisera obligatoirement la fonction précédente.

Exemple

L'appel affiche_triangle(7) devra afficher

```
76543210
6543210
543210
43210
3210
210
10
0
```

Collier d'enchère et en noce

1. Écrire une fonction collier qui prend en paramètre un entier p qui correspond à un nombre de perles et qui affiche un collier de longueur $3p + 2$ où les perles sont représentées par des caractères 'o' et le fil est représenté par le caractère '-'. Le collier devra commencer par deux morceaux de fil, puis par une perle, puis par deux morceaux de fil... et ainsi de suite jusqu'à la fin.
   Par exemple, la fonction collier(3) doit afficher : (mais ne doit rien renvoyer)

```
--o--o--o—
```

2. Écrire une fonction nombre_zeros_fin qui prend en paramètre un entier n et qui renvoie le nombre de zéros à la fin du nombre n.
   Par exemple, nombre_zeros_fin(1025000) doit renvoyer 3 alors que nombre_zeros_fin(150001) doit renvoyer 0.
   Par convention, le nombre 0 n'a qu'un seul zéro.
3. Écrire un programme offre_utilisateur(nb) qui demande à l'utilisateur un nombre avec au moins nb zéros à la fin. Si l'utilisateur ne donne pas un tel nombre, le programme lui redemandera un nombre jusqu'à qu'il rentre un nombre licite. La fonction renverra alors ce nombre.
   On utilisera obligatoirement la fonction précédente.
   Par exemple, offre_utilisateur(3) pourra donner lieu à l'échange

```
Veuillez me proposer un nombre avec 3 zéros a la fin :
14200
Vous rigolez, ce nombre n'a que 2 zéros !
Veuillez me proposer un nombre avec 3 zéros a la fin :
10001
Vous rigolez, ce nombre n'a que 0 zéros !
Veuillez me proposer un nombre avec 3 zéros a la fin :
101000
```

4. On simulera une vente en enchère d'un collier très cher pour un cadeau de mariage (ce dernier point est juste pour justifier le jeu de mots du titre) dans une fonction mise_enchere_collier qui prend en paramètre un entier p qui est le nombre de perles du collier. Tout d'abord, la fonction affichera le collier à vendre. Ensuite, deux acheteurs potentiels, nommés "1" et "2", se disputent l'achat du collier. Acheteur 1 commence à faire une offre. Petite particularité : chacune de leurs offres doit contenir au moins p zéros à la fin de leurs montants, où p est le nombre de perles. Tant qu'un acheteur augmente l'offre de l'autre, l'enchère continue. L'enchère s'arrêtera donc quand un acheteur propose une offre inférieure à celle de l'autre. La fonction renverra le numéro de l'acheteur qui a remporté le collier.

Un exemple sur mise_enchere_collier(5) :

```
Bienvenue sur l'enchere de ce magnifique collier, parfait pour un cadeau de
mariage :
--o--o--o--o--o--
Les offres commencent ! Seuls les montants avec 5 zeros a la fin sont acceptes.
Acheteur 1 !
Veuillez me proposer un nombre avec 5 zeros a la fin :
200000
Acheteur 2 !
Veuillez me proposer un nombre avec 5 zeros a la fin :
250000
Vous rigolez, ce nombre n'a que 4 zeros !
Veuillez me proposer un nombre avec 5 zeros a la fin :
300000
Acheteur 1 !
Veuillez me proposer un nombre avec 5 zeros a la fin :
2000000
Acheteur 2 !
Veuillez me proposer un nombre avec 5 zeros a la fin :
1800000
Acheteur 1 a propose la meilleure offre, c'est lui qui repart avec le collier !
```

Usine à nombres aléatoires

Petits rappels sur les nombres aléatoires en C

Une seule fonction existe en C pour générer un nombre aléatoire : il s'agit de la fonction rand() qui ne prend aucun paramètre et qui renvoie un nombre (pseudo-)aléatoire entre 0 et un entier dénoté RAND_MAX (qui vaut généralement 2147483647 - mais ça peut dépendre du compilateur/de l'architecture de la machine). Cette fonction se trouve dans la bibliothèque stdlib.h.

Les nombres sont pseudo-aléatoires, ce qui veut dire que tout l'aléatoire provient d'un nombre appelé graine (voir le cours n°5). Pour que les nombres aléatoires changent à chaque exécution du programme, on réinitialise la graine au démarrage de ce dit programme, généralement en se basant sur l'heure en cours. Concrètement, à chaque fois qu'on souhaite utiliser des nombres aléatoires, on insérera au début de votre fonction main la ligne de commande :

```
srand(time(NULL))
```

Votre fonction main devrait ressembler à ceci :

```
/* Ecrivez votre programme ci-dessous. */
#include <stdio.h>
#include <stdlib.h>
#include <time.h> // l'inclusion de cette obligatoire


int main() {
```

```
  srand(time(NULL)) // réinitialiser la graine de démarrage;

  /* Le reste du code ci-dessous*/

  return EXIT_SUCCESS;
}
```

Questions

1. Écrire une fonction flottant_aleatoire qui étant donnés deux flottants debut et fin renvoie un nombre flottant aléatoire entre debut et fin.

   Astuce : on peut avoir un flottant aléatoire x entre 0 et 1 en divisant rand() par RAND_MAX. (**Rappel.** La division de flottants de a par b peut s'effectuer en écrivant a / (float) b .) À vous de trouver ce que valent A et B pour que A*x+B soit un nombre entre debut et fin.
2. Écrire une fonction entier_aleatoire qui étant donnés deux entiers debut et fin renvoie un nombre entier aléatoire entre debut et fin (bornes comprises). Astuce : on peut avoir un entier aléatoire entre 0 et n-1 en effectuant rand() % n .

Gardez ces deux fonctions sous le coude, elles vous seront très utiles !

Est-ce un rectangle ?

1. Dans un premier temps, écrire une fonction dist_au_carre qui prend en argument quatre entiers xA, yA, xB, yB et qui calcule la distance au carré entre le point (xA, yA) et le point (xB, yB) . Pour rappel, la formule pour la distance au carré entre deux points et est donnée par : $AB^2 = (x_B - x_A)^2 + (y_B - y_A)^2$.
2. Dans un second temps, écrire une fonction angle_droit qui prend en argument six nombres entiers xA, yA, xB, yB, xC, yC, et qui renvoie 1 si les points (xA, yA), (xB, yB), et (xC, yC) forment un angle droit en (xB, yB), 0 dans le cas contraire. On utilisera pour cela le théorème de Pythagore qui dit que l'angle$\widehat{ABC}$ forme un angle droit si et seulement si $AB^2 + BC^2 = AC^2$. La fonction angle_droit fera donc appel à la fonction dist_au_carre.
3. Enfin, écrire une fonction rectangle qui prend en argument huit nombres entiers xA, yA, xB, yB, xC, yC, xD, yD et qui renvoie 1 si les points (xA, yA), (xB, yB), et (xC, yC) et (xD, yD) forment un rectangle (dans cet ordre), 0 sinon. Par exemple, rectangle(0,1,2,0,4,4,2,5) doit renvoyer 1, mais rectangle(0,1,2,0,2,3,4,-1) doit renvoyer 0.

Grand bien te face (ou grand bien te pile)

1. Définissez deux macros constantes PILE et FACE valant respectivement 0 et 1.
2. Écrivez une fonction pile_ou_face sans argument qui renvoie PILE avec 50% de chance et FACE avec 50% de chance.
3. Écrivez une fonction proportion_piles, prenant un entier en argument, qui effectue autant de pile ou face que cet entier et qui renvoie la proportion de piles par rapport au nombre de lancers total.
   Par exemple, si on réalise les pile ou face FACE FACE PILE FACE FACE PILE FACE PILE, alors proportion_piles(8) renvoie 0.375 (ce qui correspond à la fraction 3/8).
4. Écrivez une fonction piles_consecutifs, prenant un entier en argument, qui simule une succession de pile ou face (grâce à la fonction pile_ou_face), s'arrête quand on tire à la suite un nombre de piles égal à l'entier en paramètre et qui renvoie le nombre de pile ou face qui ont été lancés jusqu'alors.
   Par exemple, si on réalise les pile ou face FACE FACE PILE FACE PILE PILE PILE, alors piles_consecutifs(3) renvoie 7 car entre le lancer 5 et le lancer 7, on est tombés sur 3 piles consécutifs.

5. Écrivez une fonction nombre_minimum_piles_et_faces, prenant un entier n en argument, qui simule une succession de pile ou face, s'arrête quand on a réalisé au moins n piles et au moins n faces, et qui renvoie le nombre de pile ou face qui a été lancé jusqu'alors.
   Par exemple, si on réalise les pile ou face FACE FACE PILE FACE FACE FACE PILE PILE, alors nombre_minimum_piles_et_faces(3) renvoie 8 car on n'a obtenu 3 piles qu'au huitième lancer (le nombre de 3 faces a été atteint au bout du quatrième).