Examen de Programmation C (session 2)

Mardi 19 juin 2018, de 10h30 à 12h30

Exercice 1. Somme et signe du produit de 2 entiers — 2 *pt*

Soient a et b deux entiers.

► **1.** Écrire un programme qui lit a et b, puis détermine et affiche la somme $a + b$.

► **2.** Modifier le programme pour également afficher le signe du produit $a \times b$, sans le calculer explicitement.

Exercice 2. Somme des entiers de 1 à n — 3 *pt*

Soit un entier $n > 0$. La somme S des entiers de 1 à n est définie de la manière suivante :

$$S = \frac{n \cdot (n+1)}{2}.$$

► **1.** Écrire un programme qui lit une valeur entière n, puis qui calcule et affiche la somme S, en utilisant la formule précédente.

► **2.** Modifier le programme pour calculer cette somme en utilisant une boucle **`for`**.

► **3.** Modifier à nouveau le programme pour utiliser une boucle **`while`**.

Exercice 3. Nombre parfait — 3 *pt*

Un nombre parfait est un nombre n qui est égal à la somme de ses diviseurs stricts (c'est-à-dire, n exclu).

► **1.** L'entier 28 est-il un nombre parfait ? Justifier la réponse.

► **2.** Écrire un programme qui lit un nombre entier n au clavier, puis qui affiche ses diviseurs.

► **3.** Modifier le programme pour déterminer si ce nombre entier n est un nombre parfait.

Exercice 4. Ré-inventons la division — 2 *pt*

Soit $x > 0$ un nombre flottant. Le résultat de $1/x$ peut être calculer en utilisant la suite définie de la manière suivante :

$$r_{i+1} = r_i \cdot (2 - x \cdot r_i), \quad \text{avec } r_0 \text{ un nombre flottant donné.}$$

Plus particulièrement, si r_0 est dans l'intervalle $]0, 2/x[$, la suite r_i converge vers $1/x$.

► **1.** Déterminer r_3 pour $(x, r_0) = (3, 0.5)$.

► **2.** Écrire un programme qui lit les valeurs flottantes (x, r_0) et un entier n au clavier puis, si r_0 est valide, calcule et affiche les valeurs r_i pour $i \in \{0, \cdots, n\}$.

Exercice 5. Boucle imbriquée et fonction — 2 *pt*

► **1.** Écrire une fonction qui prend une valeur entière n en paramètre, puis qui affiche un motif comme celui ci-dessous pour $n = 2$.

```
XXXXXX  XXXX  XXXX
XXXXXX  XXXX  XXXX
XX  XX  XX  XX  XX
XX  XX  XX  XX  XX
XXXXXX  XX      XX
XXXXXX  XX      XX
```

Exercice 6. Valeurs min-max d'un tableau 2 *pt*

Soit t un tableau d'entiers de taille 20.

► **1.** Écrire un programme qui lit le contenu de t, puis qui détermine en un seul parcours et affiche la valeur $\min$ et la valeur $\max$ du tableau t.

Exercice 7. Échange de valeurs 2 *pt*

Soit la fonction `foo` suivante, qui est censée faire l'échange entre deux valeurs entières a et b. Or quand j'exécute ce programme, aucune des deux valeurs n'est modifiées en sortie de la fonction.

```
void foo(int a, int b) {
  a = a + b;
  b = a - b;
  a = a - b;
}

int main(void) {
  int a = 17, b = 18;
  foo(a, b);
  printf("a= %d et b= %d \n", a, b);
  return 0;
}
```

► **1.** Proposer une explication à ce problème.

► **2.** Modifier la fontion `foo` pour qu'elle effectue effectivement l'échange entre a et b.

Exercice 8. Rotation des éléments d'un tableau 4 *pt*

Soit t un tableau de taille n. On souhaite effectuer une rotation de k rang vers la gauche des éléments de t.

► **1.** Écrire une fonction qui prend en paramètre le tableau t et sa taille n, et un indice k, et qui effectue une rotation de k rang vers la gauche des éléments de t, en utilisant les crochets (`[]`) pour accéder aux éléments du tableau.

Par exemple, si $t = \{0, 1, 2, 3, 4\}$ et $(n, k) = (5, 2)$, après rotation, on aura $t = \{2, 3, 4, 0, 1\}$.

► **2.** Écrire une programme principal qui permet d'utiliser cette fonction sur un tableau de taille 20 initialisé à la déclaration avec les valeurs $1, 2, \cdots, 20$.

► **3.** Modifier la fonction précédente pour accèder aux éléments du tableau en utilisant l'arithmétique des pointeurs.

Exercice 9. Carré magique 4 *pt*

Un *carré magique* d'ordre n est une grille de taille $n \times n$ composée de nombres entiers, telle que la somme des éléments de chaque ligne, de chaque colonne, et de chaque diagonale sont égales : cette somme est la *densité* du carré magique. Par exemple, la grille suivante est un carrée magique de densité 15.

2	7	6
9	5	1
4	3	8

En effet $(2+7+6) = (9+5+1) = (4+3+8) = (2+9+4) = (7+5+3) = (6+1+8) = (2+5+8) = (4+5+6) = 15$.

► **1.** Écrire une fonction qui prend une grille g de taille 3×3 en paramètre, qui détermine si cette grille est un carré magique, et affiche sa densité (ou -1).

► **2.** Écrire un programme principal qui permet d'utiliser cette fonction.

► **3.** Que faut-il modifier (fonction et programme principal) pour pouvoir effectuer ce même traitement sur une grille de taille $n \times n$, avec n une valeur entière lue au clavier.