Programmation en C et structures de données

guillaume.revy@univ-perp.fr

Fonctions et pointeurs

Exercice 1. Périmètre d'un cercle

► **1.** Écrire une fonction qui calcule et retourne le périmètre d'un cercle de rayon r passé en paramètre.

► **2.** Écrire un programme principal pour tester cette fonction.

Exercice 2. Calcul de distance

► **1.** Écrire une fonction qui prend en paramètre 4 flottants (**`double`**) qui représentent les coordonnées de deux points A et B, puis qui calcule et retourne la distance AB.

► **2.** Écrire un programme principal qui permet de tester cette fonction sur 4 flottants (**`double`**) lus au clavier.

Exercice 3. Affichage de la table ASCII

En langage C, un caractère, c'est-à-dire, une variable de type **`char`** ou **`unsigned char`**, est représenté en mémoire par un entier sur 8 bits (1 octet), interprété *à l'affichage* comme un caractère selon le codage ASCII par l'utilisation de `%c`. En réalité, chaque caractère d'un texte en ASCII est stocké dans un octet dont le bit de poids 7 (8^e bit) est 0. La table ASCII ne permet donc de représenter que 128 caractères.

► **1.** Écrire une fonction qui retourne le caractère associé à un nombre entier (**`unsigned short`**) passé en paramètre.

► **2.** Utiliser cette fonction pour afficher les 128 caractères de la table ASCII.

► **3.** Écrire ensuite une fonction qui retourne l'entier associé à un caractère passé en paramètre.

Exercice 4. Un peu de ASCII art

► **1.** Écrire une fonction qui prend en paramètre un entier n, et qui affiche une ligne de n caractères '*', suivie d'un retour à la ligne.

► **2.** Utiliser cette fonction pour afficher un carré de caractères '*' de taille $n \times n$.

Exercice 5. Un drôle de damier

Ici un damier carré est caractérisé par deux paramètres : un nombre t de cases sur chaque ligne et la taille c d'une case. Pour $(t, c) = (4, 2)$, le damier aura la forme suivante.

```
XX..XX..
XX..XX..
..XX..XX
..XX..XX
XX..XX..
XX..XX..
..XX..XX
..XX..XX
```

► **1.** Écrire une fonction qui prend en entrée les paramètres t et c, puis dessine le damier correspondant.

► **2.** Écrire un programme principal pour tester cette fonction.

Exercice 6. Un drôle de fermier

Un fermier fait l'élevage de moutons et de dindons. Au moment de payer ses impôts, il déclare qu'il a dans son élevage t têtes et p pattes.

- ▶ **1.** Sachant qu'un mouton et un dindon ont chacun 1 tête, que le mouton a 4 pattes et que le dindon n'en a que 2, écrire une fonction qui prend en paramètre t et p, puis détermine de manière itérative et affiche le nombre m de moutons et d de dindons de l'élevage (et si ce n'est pas possible, l'indique à l'utilisateur).
- ▶ **2.** Écrire un programme principal pour tester votre fonction avec $t = 36$ et $p = 100$.
- ▶ **3.** Écrire un programme principal qui permet de saisir la valeur de t et p au clavier, puis affiche le nombre de moutons et de dindons correspondant.

Exercice 7. Fonctions sur les tableaux d'entiers

- ▶ **1.** Écrire une fonction `min_tab` qui retourne la valeur minimum d'un tableau d'entiers passé en paramètre. Modifier votre fonction pour retourner la valeur maximale.
- ▶ **2.** Écrire une fonction `moyenne_tab` qui retourne la moyenne des valeurs d'un tableau d'entiers passé en paramètre.
- ▶ **3.** Écrire une fonction qui prend en paramètre un entier n, et retourne le nombre de valeurs du tableau plus petites que n.
- ▶ **4.** Écrire un programme principal, permettant de tester ces fonctions.

Exercice 8. Un algorithme de tri

Le tri à bulle consiste à comparer répétitivement les éléments consécutifs d'un tableau, et à les permuter lorsqu'ils sont mal triés. [1]

- ▶ **1.** Écrire une fonction qui prend un tableau d'entiers de taille n en paramètre, et qui affiche son contenu sur la sortie standard.
- ▶ **2.** Écrire une fonction qui tire aléatoirement les éléments d'un tableau de taille n passé en paramètre.
- ▶ **3.** Écrire enfin une fonction qui trie de manière décroissante en utilisant le tri à bulle un tableau de taille n.
- ▶ **4.** Proposer un programme qui permette de tester l'ensemble de ces fonctions.

Exercice 9. Pointeurs

- ▶ **1.** Écrire un programme qui réalise les actions suivantes :
  1. déclarer un entier i et un pointeur vers un entier p,
  2. initialiser l'entier à une valeur arbitraire, et faire pointer p vers i,
  3. afficher la valeur de i,
  4. modifier la valeur pointée par p, et afficher la valeur de i,
  5. et afficher l'adresse de i, la valeur de p et celle pointée par p.

Exercice 10. Pointeurs et allocation dynamique

- ▶ **1.** Écrire un programme qui lit un entier n au clavier, puis alloue un tableau de flottants (**`float`**) de taille n et le remplit de manière aléatoire, avant de l'afficher.
- ▶ **2.** Modifier le programme pour trier ce tableau en utilisant l'algorithme de tri à bulles (vu précédemment), sans utiliser l'opérateur d'accès `[]` mais uniquement l'arithmétique des pointeurs pour parcourir le tableau.

Exercice 11. Fonction d'échange

- ▶ **1.** Écrire une fonction qui prend en paramètre deux nombres entiers, et qui effectue l'échange de ces deux valeurs.
- ▶ **2.** Écrire un programme principal qui lit deux valeurs entières au clavier, et permet d'illustrer l'utilisation de cette fonction d'échange.

1. Pour plus de détails : `https://fr.wikipedia.org/wiki/Tri_à_bulles`.

Exercice 12. Fonction, tableau et structure

On souhaite stocker les dates de naissances des tous les étudiants de L2 Informatique dans un tableau. On utilisera un tableau statique de taille 24.

- **1.** Définir une structure `une_date` contenant les attributs entiers suivant, dans cet ordre : le jour (**`int`**), le mois (**`int`**), et l'année (**`int`**).
- **2.** Écrire un programme principal qui permet de créer une date, puis de lire et afficher les différents champs d'une date.
- **3.** Étendre ce programme pour lire les 24 dates et les stocker dans un tableau.
- **4.** Écrire une fonction qui prend en paramètre le tableau de dates puis les affiches, dans l'ordre chronologique.