Algorithmique et Programmation 1
IMAC 1ere année

TP 3

Itérations, fonctions, tableaux

Dans cette séance de travaux dirigés, les points suivants sont abordés :
— les boucles : `while`, `for`, `do while` ;
— les fonctions ;
— les tableaux ;
— la manipulation basique de chaînes de caractères.

Pendant le TP, vous écrirez vos codes dans des fichiers source. Ayez le réflexe d'enregistrer régulièrement vos sources. N'hésitez pas à créer un nouveau fichier pour les (sous-)exercices qui démandent de modifier le code précédent. Ça vous aidera plus tard pendant le temps de révision.

À la fin de la séance, mettez-les sources dans une archive (commande `tar -zcvf nom_archive fichiers_source`) et rendez-le suivant les instructions données sur e-learning. N'oubliez pas de commenter votre code.

Pensez à consulter les transparents de cours sur elearning régulièrement.

Tous les exercices de cette feuille devraient être faits par tout le monde ; il n'y a pas d'exercices 'bonus'. La feuille risque d'être trop longue pour la séance, il faut donc faire attention aux instructions données à sa fin. Par défaut, il faudra (tenter de) finir les exercices et poser les questions éventuelles la semaine prochaine.

Exercice 1. (Itérations)

1. Écrire une boucle `while` qui affichera les entiers entre 1 et 50.
2. Écrire une boucle `while` qui affichera les 50 premiers multiples de 7, chaque sur une ligne de forme

   `5 fois 7 est 35.`

3. Refaire les deux questions précédentes en utilisant une boucle `for`.

Exercice 2. (Somme d'une suite positive)

Écrire un programme qui laisse l'utilisateur entrer des entiers positifs. Une fois un entier négatif saisi, la lecture s'arrête et le programme affiche la somme et la moyenne des nombres entrés. Le dernier nombre négatif n'est pas pris en compte pour le calcul.

Par exemple, sur l'entrée

$$4 \quad 2 \quad 0 \quad 5 \quad 1 \quad -4,$$

le programme affiche `somme = 12, moyenne = 2.4`.

Exercice 3. (Fonctions : les bases)

Cet exercice introduit la notion de fonctions dans sa forme la plus basique (oublier pour le moment les adresses et pointeurs ; on les traitera la prochaine fois).

1. Écrire une fonction `affiche_entiers` qui prend en paramètre un entier `n` et qui affiche tous les entiers entre 1 et `n`. La tester dans un programme qui laisse l'utilisateur choisir ce nombre.
2. Écrire une fonction `ajoute_3` qui prend en paramètre un entier et qui renvoie (sans rien afficher) son valeur augmenté de 3. Tester dans un programme qui démande l'utilisateur de saisir un entier et qui affiche cet entier augmenté de 3.

REMARQUE : On demande bien que la fonction `ajoute_3` renvoie une valeur qui peut être ensuite utilisée à volonté. C'est le comportement auquel on s'attendra par défaut quand on demandera une fonction qui performe un calcul ou une opération quelconque sauf si on demande exprès d'afficher le résultat.

Exercice 4. (Tableaux)

1. Se familiariser avec le comportement basique des tableaux :
   — déclarer un tableau `tab1` de 5 entiers contenant les nombres 6, 7, 8, 9, 10 ;
   — afficher l'élément `tab1[3]` : à quelle case corresponde-t-il ?
   — modifier la valeur de cet élément et l'afficher de nouveau ;
   — afficher `tab1[5]` et expliquer sa valeur.
2. Définir une constante `MAX_TAILLE` et l'utiliser dès maintenant pour spécifier la taille des tableaux créés (super important pour pouvoir simplement modifier le code plus tard et pour éviter plein d'erreurs assez dangereux et difficiles à traquer).
3. Vérifier qu'on ne peut pas traiter les tableaux comme les vairables 'normaux' :
   — tenter d'afficher le tableau `tab1` d'une manière naïve, e.g. `printf("%d", tab1)` ;
   — déclarer un autre tableau d'entiers, `tab2`, de la même taille que `tab1` ;
   — tenter d'y affecter une valeur avec `tab2 = tab1` ou bien `tab2 = {1, 2, 3, 4, 5}`.

Exercice 5. (Manipulation des tableaux)

En général les tableaux n'admettent pas d'opérations globales. Pour manipuler un tableau, il faut le parcourir et traiter les éléments un par un.

1. Écrire une boucle qui parcourt un tableau d'entiers de taille `MAX_TAILLE` et qui affiche chaque de ses éléments. Tester.
2. À sa base, écrire une fonction `affiche_tab` qui prend en paramètre un tableau d'entiers et un entier `taille` et qui affiche les premiers `taille` éléments de ce tableau. Tester.
3. Écrire une fonction `remplir_tab` qui prend en paramètre un tableau d'entiers et un entier `taille` et qui laisse l'utilisateur remplir les `taille` premiers cases de ce tableau. Tester.
4. Écrire une fonction `copy_tab` qui prend en paramètre deux tableaux d'entiers et leur taille effective et qui copie le contenu du deuxième tableau dans le premier. Tester.

5. Écrire une fonction `somme_tab` qui prend en paramètre un tableau d'entiers et sa taille effective et qui renvoie la somme d'éléments de ce tableau. Tester.

Exercice 6. (Tableaux de caractères)

Si on sait traiter les tableaux d'entiers, on sait également traiter les tableaux de caractères. Pour déclarer et initialiser un tableau de caractères :

`char tab[MAX_TAILLE] = {'H', 'e', 'l', 'l', 'o'}`.

Les caractères peuvent être affichés et scannés comme les entiers en remplaçant les `"%d"` par `"%c"`.

1. Modifier la fonction `affiche_tab` pour qu'elle affiche les tableaux de caractères. Tester.
2. Modifier la fonction `remplir_tab` pour qu'elle remplit les tableaux de caractères. Tester. Noter que les espaces et sauts de ligne sont cette fois aussi enregistrés. Pour l'éviter, il suffit de remplacer `"%c"` dans scanf par `" %c"` (un espace de plus).

Exercice 7. (Chaînes de caractères)

Si on sait traiter les tableaux de caractères, on sait aussi travailler avec les chaînes de caractères. Une chaîne de caractères est un tableau de caractères qui est terminé avec un symbol spécial noté 0 ou '\0' (pas la même chose que '0' !).

Il n'y a aucune différence entre

`char tab[MAX_TAILLE] = "Hello"`

et

`char tab[MAX_TAILLE] = {'H', 'e', 'l', 'l', 'o', '\0'}`.

Le zéro à la fin enlève la nécessité de stocker la longueur de la chaîne dans une variable séparée : on peut toujours tester si on a déjà atteint la fin ou pas encore.

1. Écrire une fonction `affiche_chaine` qui affiche une chaîne de caractères donnée en paramètre (et dont la longueur n'est pas spécifiée).
2. Écrire une fonction `longueur_chaine` qui renvoie la longueur d'une chaîne de caractères reçue en paramètre.
3. Tester `printf("%s")` et `scanf("%s")` qui servent à manipuler les chaînes de caractères.