Algorithmique et Programmation 1
IMAC 1ere année

TP 5

Chaînes de caractères, fonctions récursives

Dans cette séance de travaux dirigés, les points suivants sont abordés :
— les chaînes de caractères ;
— les fonctions récursives.

Comme lors des séances précédentes, pensez à faire régulièrement des sauvegardes (dans des fichiers distincts) et à documenter votre code. Pensez à toujours tester votre code. 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.

Pensez à consulter les transparents de cours sur elearning régulièrement.

Exercice 0. (Avant de commencer)

Poser de questions s'il reste quelque chose de peu clair sur les pointeurs et leur utilisation dans les fonctions. C'est un concept particulièrement important.

Exercice 1. (ASCII)

1. Se familiariser avec la notion de caractère et de son code ascii :
   — déclarer et initialiser une variable `char lettre = 't'` ;
   — l'afficher en tant qu'un caractère et qu'un entier (`%c` et `%d`) ;
   — augmenter `lettre` par 2 et répéter l'affichage.
2. Écrire un programme qui affiche les caractères de 'A' jusqu'à 'Z' et de '0' jusqu'à '9' suivis par la valeur de leur code ASCII.

Exercice 2. (Chaînes de caractères)

Une chaîne de caractères est un tableau de caractères terminé avec un caractère spécial noté 0 ou '\0' (pas la même chose que '0' !).

Il n'y a pas de différence entre

```
char tab[MAX_TAILLE] = "Hello"
```

et

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

Le zéro permet de détecter la fin d'une chaîne et enlève donc la nécessité de stocker l'information sur sa longueur.

1. Écrire une fonction `void affiche_chaine(char chaine[])` qui affiche une chaîne de caractères donnée en paramètre. Tester.
2. Tester `printf("%s", ...)` et `scanf("%s", ...)` qui permettent d'afficher et de saisir les chaînes de caractères.

Exercice 3. (Traitement des chaînes)

1. Écrire une fonction `longueur_chaine` qui renvoie la longueur d'une chaîne de caractères reçue en paramètre.
2. Écrire une fonction `void copie_chaine(char dest[], char src[])` qui copie la chaîne `src` en `dest` (présumons que `dest` offre assez d'espace).
3. Tester les fonctions `strlen` et `strcpy` mises à disposition par `string.h` qui servent au même but.
4. Écrire une fonction `int compte_chiffres(char chaine[])` qui compte le nombre de chiffres contenus dans une chaîne. E.g. pour la chaîne `"ch41n3"`, la fonction renvoie 3.

Exercice 4. (Fonctions récursives)

1. Écrire une fonction récursive, `int fact(int n)` qui permet de calculer la factorielle d'un entier positif passé en paramètre.
2. Écrire une fonction récursive `pgcd_euc` qui renvoie le PGCD de 2 entiers strictement positifs A et B en utilisant l'algorithme d'Euclide qui s'appuie sur les propriétés suivantes :

   $PGCD(A, B) = B$ si B est un diviseur de A,
   $PGCD(A, B) = PGCD(B, A \mod B)$ sinon.

   Donc par exemple $PGCD(36, 20) = PGCD(20, 16) = PGCD(16, 4) = 4$.

Exercice 5. (Récursion mutuelle)

Les fonctions récursives peuvent parfois être utiles pour résoudre directement un problème sans devoir l'analyser et reformuler. Soient les suites U et V définies par

$$U_0 = 2\,, \qquad V_n = \frac{2}{U_n}\,, \qquad U_{n+1} = \frac{U_n + V_n}{2}\,.$$

1. Écrire deux fonctions `double suiteU(int n)` et `double suiteV(int n)` qui calculent les valeurs de U_n et V_n pour un entier positif n. (Implémenter en suivant tout bêtement la définition des suites : les fonctions feront appel l'une à l'autre dans leur calcul). Tester avec de petits n.
2. Combien d'appels de ces fonctions sont effectués pour calculer la valeur de U_5 ? Et de U_{50} ?
3. Faire la substitution de V_n par $\frac{2}{U_n}$ dans le calcul de U_{n+1}. Modifier la fonction `suiteU()` en corréspondance. Combien appels faut-il pour calculer U_{50} maintenant ?

Pour s'exercer

Exercice 6. (Calcul puissance)

1. Écrire une fonction récursive qui prend en paramètre deux entiers positifs, n, p, et calcule n^p. Combien de fois sera-t-elle appelé dans un calcul de n^p ?
2. Si c'est aux allentours de p fois, on peut faire largement mieux en exploitant le fait que pour p pair, $n^p = \left(n^{p/2}\right)^2$. Par exemple $n^{13} = n \times n^{12}$, $n^{12} = (n^6)^2$, $n^6 = (n^3)^2$, ... Ainsi, on peut calculer la puissance en $\log p$ appels de la fonction. Modifier la fonction pour utiliser cette methode de calcul.

Exercice 7. (Tableaux – suite)

Choisir une methode de renvoi de résultats approprié pour chaque des fonctions demandées.

1. Écrire une fonction `min_max_tab()` qui calcule la valeur minimale et la valeur maximale d'un tableau d'entiers saisi en paramètre.
2. Écrire une fonction `cherche_entier()` qui permet de chercher un entier donné dans un tableau d'entiers et qui renvoie l'index de sa première occurence. En cas d'échec, la valeur renvoyé sera -1.

Exercice 8. (Chaînes de caractères – suite)

1. Écrire une fonction `void inverse_chaine(char[] chaine)` qui inverse la chaîne de caractères fournie en paramètre. Par exemple `"premier"` deviendra `"reimerp"`.
2. Écrire une fonction `void enleve_majuscules(char[] chaine)` qui enlève toutes les majuscules ('A'-'Z') de la chaîne fournie. Par exemple `"PreMieR"` deviendra `"reie"`.

Exercice 9. (string.h)

1. Récherche en ligne : regarder les fonctions mises en disposition par `string.h` (à travers, pas besoin de mémoriser)
2. Écrire une fonction avec la même fonctionnalité que `strcmp` par propres soins.