

Introduction à la programmation

TD 4

Polytech Marseille - 3A

Filière Génie Biomédical

Exercice (Challenge).

(i) *Ecrire une fonction :*

```
int occurrences (float * tab, int size_tab, float x)
```

renvoyant le nombre d'occurrences (d'apparitions) de `x` dans le tableau `tab` de taille `size_tab`.

(ii) *Pourquoi peut-on utiliser le type `float *` pour désigner un tableau de réels ?*

(iii) *Ecrire un main permettant de tester la fonction sur un tableau contenant (1.1, 3.1, 2, 4.3, 2, 1.1) avec `x1 = 1.1`, `x2 = 2`, `x3 = 5`.*

(iv) *Ecrire une fonction :*

```
float * elements_tableau (float * tab, float x, int *pnb_vals)
```

créant dynamiquement (de la bonne taille) et retournant un tableau contenant une et une seule fois chaque élément du tableau `tab` et stockant dans l'entier d'adresse `pnb_vals` le nombre de cases du tableau retourné. Bien sûr, votre fonction devra être la plus rapide possible !

Pour l'exemple précédent, il faudra retourner allouer un tableau de taille 4 contenant les valeurs (1.1, 3.1, 2, 4.3) (peu importe l'ordre) et l'entier stocké à l'adresse `pnb_vals` recevra 4.

1 Pointeurs

Exercice 1. Ecrire un programme déclarant une variable numérique `v` et l'initialisant. Déclarer un pointeur de type correspondant à cette variable et le faire pointer sur `v`. Modifier la valeur de `v` indirectement en utilisant le pointeur. Contrôler l'adresse et la valeur des variables au fur et à mesure des différentes instructions au moyen de `printf`.

Exercice 2. Ecrire une fonction prenant en argument deux variables réelles et échangeant leur contenu.

Exercice 3. Ecrire une fonction prenant en argument trois nombres réels `x`, `y` et `z` et ordonnant leur contenu (donc à la sortie de la fonction `x` contiendra la plus petite des trois valeurs, `y` la seconde, etc...).

Exercice 4. Ecrire une fonction :

```
float racine (float x, int *err)
```

renvoyant la racine de `x` quand elle existe et utilisant la variable `err` comme code d'erreur :

- si la racine de `x` n'existe pas, mettre `err` à `-1`
- si le calcul s'est bien passé, la mettre à `0`

Exercice 5 (Recherche d'un élément dans un tableau). Ecrire une fonction :

```
int recherche (float x, float tab[], int *err)
```

recherchant la valeur `x` dans le tableau `tab` et renvoyant la position de cette dernière si elle est trouvée. Comme dans la question précédente, on utilisera `err` comme code d'erreur (avec les mêmes conventions : -1 si la variable n'est pas trouvée, 0 sinon).

Modifier la fonction pour obtenir une fonction :

```
float * recherche_pt (float x, float tab[], int *err)
```

renvoyant non plus le numéro de la case dans le tableau mais un pointeur sur la valeur trouvée.

Pour mettre en pratique, dans le `main`, déclarer et remplir un tableau (de manière statique), puis faire saisir à l'utilisateur une succession de valeurs réelles (on arrêtera la saisie quand -1 est rencontré) et les rechercher dans le tableau (on affichera à l'utilisateur si la valeur a été trouvée ou non et à quelle position). Voyez-vous l'intérêt de la variable `err`?

2 Pointeurs et chaînes de caractères

Exercice 6. Ecrivez la fonction

```
int strlen(char *s);
```

qui détermine et renvoie le nombre de caractères d'une chaîne de caractères.

Exercice 7. Ecrivez la fonction

```
int strcmp(char *s1, char *s2);
```

définie par :

$$\text{strcmp}(a, b) = \begin{cases} \text{une valeur négative} & \text{si } a < b \\ 0 & \text{si } a = b \\ \text{une valeur positive} & \text{si } a > b \end{cases}$$

l'ordre sur les chaînes étant l'ordre lexicographique habituel.

Exercice 8. Ecrivez la fonction

```
char *strcpy(char *dest, char *srce);
```

qui copie la chaîne `srce` dans l'espace pointé par `dest` et rend comme résultat l'adresse de cet espace.

Exercice 9. Ecrivez la fonction

```
char *strcat(char *s1, char *s2);
```

qui rend la chaîne `s1` concaténée avec `s2` (`s1` et `s2` sont mises bout à bout), sans recopie de `s1`. Quel avertissement doit-on faire aux utilisateurs de cette fonction ? Illustrez votre réponse par une mauvaise et une bonne utilisation de `strcat`.

Exercice 10. Une chaîne de caractères est un *palindrome* si elle a moins de deux caractères ou si son premier et son dernier caractères sont identiques et la sous-chaîne obtenue en les retirant est elle aussi un palindrome :

esoperesteicietserepose.

Ecrivez la fonction

```
int palindrome(char *s, int debut, int fin)
```

qui rend 1 si la suite de caractères $s_{debut}, s_{debut+1}, \dots, s_{fin}$ constitue un palindrome, 0 sinon.

3 Malloc

Exercice 11.

- (i) Déclarer deux variables : n de type entier, et $ptab$ de type pointeur sur un flottant.
- (ii) Faire saisir à l'utilisateur la valeur de n .
- (iii) Allouer un tableau de taille n dont on stockera l'adresse dans $ptab$.
- (iv) Remplir le tableau en stockant dans la case i la valeur $5 * i$.
- (v) Afficher le tableau obtenu.

Exercice 12.

On déclare les variables suivantes :

```
char buffer[20] ;  
char *palin ;
```

- (i) Faire saisir par l'utilisateur un mot dans `buffer` (longueur maximale 20 caractères, pour cela, on utilisera la fonction `fgets`, taper `man fgets` en console pour une aide - la saisie clavier correspond au flux nommé `stdin`).
- (ii) Allouer dans `palin` un tableau de taille appropriée pour stocker le palindrome dont la première moitié est celle tapée par l'utilisateur dans `buffer`.
- (iii) Créer dans `palin` le palindrome correspondant.
- (iv) Utiliser la fonction `palindrome` précédente pour tester qu'il s'agit bien d'un palindrome.

Par exemple, si l'utilisateur saisit `azerty` dans `buffer`, on créera le palindrome : `azertytreza` dans `palin`.

4 Bonus

Exercice 13.

- (i) Définir un type `struct stat` contenant les champs suivants :
 - `sum`, `sum2`, `n`, `mean`, `stddev` tous de type `double`
- (ii) Etant donné un tableau de doubles `t`, écrire une fonction :

```
void stat_parcours(double t[], struct stat s)
```

mettant à jour dans `s` les champs `sum`, `sum2` et `n` contenant respectivement le somme des éléments du tableau, la somme des carrés de ces éléments et le nombre d'éléments.

- (iii) Ecrire une fonction :

```
void maj_stat(struct stat s)
```

finissant le calcul des statistiques, c'est-à-dire celui de `mean` et `stddev`