TD2 : Pointeurs

Ne pas oublier de créer un répertoire pour placer les codes de ce TP !

Toutes les fonctions écrites doivent être testées dans le programme principal !

Exercice 1 (*Lecture de code*)
Qu'affiche le code suivant ? Vérifier en exécutant ce code sur votre ordinateur.

```
#include <stdio.h>

int main(void) {
    int a,b,c;
    int *pa, *pb;
    pa=&a;
    pb=&b;
    a=2;
    b=3;
    c=5;
    printf("%d  %d \n",*pa, *pb);
    *pa=a+c;
    *pb-=c;
    pb=&c;
    printf("%d  %d  %c \n", a , b, c);
    *pb=*pa+c;
    a=c*(*pb);
    printf("%d  %d  %d \n", a , b, c);
    return 0;
}
```

Exercice 2 (*Tableau et pointeur*)
Soit P un pointeur qui 'pointe' sur un tableau `tab` d'entiers de taille 10 :

```
int tab[10] = {17, 57, 64, 16, 99, 73, 75, 51, 21, 19};
int *P;
P = tab;
```

À quelles adresses ou valeurs correspondent les expressions suivantes ? Vérifier en exécutant ce code :

1. P
2. *(P+2)
3. *P+2
4. P+2
5. &tab[6]-&tab[2]
6. &tab[5]-P
7. P+(*P-10)
8. P+*(P*7)
9. *(P+*(P+6)-tab[5])

Exercice 3 (*Calcul d'heure*)

1. Écrire une fonction/procédure *ajouteSeconde* qui ajoutera à un temps (heure : min : s) passé en paramètre un nombre de secondes passées en paramètre. Est-il possible de faire cela avec des paramètres classiques ? Quelle est la solution ?
2. Tester la fonction dans le programme principal en faisant saisir l'heure et le nombre de secondes à lui ajouter. Afficher la nouvelle heure calculée dans le programme principal.

Exercice 4 (*Somme d'un tableau*)

— Écrire un programme qui remplit un tableau d'entiers de taille 20 avec des nombres aléatoire entre 0 et 10 et calcule la somme de ses éléments en **utilisant un pointeur pour son parcours** (vous n'avez pas le droit d'utiliser le nom du tableau lors du parcours !).

Exercice 5 (*resolution d'équation*)

1. Écrire une fonction *int resoudre1(float a, float b, float *res)* qui résout l'équation suivante : $a * x + b = 0$. La fonction retourne le nombre de solutions trouvées (1 s'il y a une solution, 0 s'il n'y en a pas, et -1 si tout x est une solution). Cette fonction modifie x pour lui donner la valeur des la solution lorsque celle-ci existe.
2. Écrire une fonction `int resoudre1(float a, float b, float c, float *res1, float *res2)` qui résout l'équation suivante : $a * x^2 + bx + c = 0$. La fonction retourne le nombre de solutions trouvées et stocke les solutions dans res1 et res2.

Exercice 6 (*Conversion de coordonnées*)

Les coordonnées cartésiennes planaires d'un point sont données par les distances `x` et `y`, respectivement l'abscisse à l'origine et l'ordonnée à l'origine.

Les coordonnées polaires d'un point dans un plan sont données par la distance `r` et l'angle `a`, respectivement la distance par rapport à l'origine (coordonnée radiale, ou rayon) et l'angle formé par le point et la demi-droite Ox représentant l'angle 0° (coordonnée angulaire ou angle polaire)

Le passage des coordonnées cartésiennes aux coordonnées polaires se fait de la manière suivante :

$$r = \sqrt{x^2 + y^2}$$
$$\theta = \begin{cases}]-\infty;\infty[& si\ r = 0 \\ atan2(y,x) & si\ r \neq 0 \end{cases}$$

Le passage des coordonnées polaires aux coordonnées cartésiennes se fait de la manière suivante :

$$x = r.cos(\theta)$$
$$y = r.sin(\theta)$$

1. Écrire une procédure `void cartesian2polar(float x, float y, float* radius, float* angle)` qui va transformer des coordonnées cartésiennes (passées par valeur) en des coordonnées polaires (retournées par adresse).
2. Écrire la procédure inverse `void polar2cartesian(float radius, float angle, float* x, float* y)` qui va permettre d'effectuer la conversion inverse.

Exercice 7 (*Recheches des extremums*)

Écrire une fonction `getExtremums(...)` qui prend en entrée un tableau `tab` de `N` valeurs numériques entières, et qui retourne un entier pour indiquer si tout s'est bien passé, ainsi que les adresses mémoire de la plus petite et de la plus grande valeur présentes dans ce tableau.

Si le but de cette fonction ne peut être réalisé (ex : le tableau est de taille 0, l'adresse du tableau est NULL, ...), la valeur de retour vaudra 1, et les valeurs des extrememums ne seront pas prises en compte par la fonction appelante.

Exercice 8 (*palindrome*)

1. Déclarer une chaîne de caractère (de taille suffisamment grande) dont l'utilisateur saisira le contenue.
2. On rappelle que la fonction `strlen()` appartenant à la bibliothèque `string.h` retourne la taille d'une chaîne de caractère passé en paramètre. Récupérer la taille de la chaîne de caractère saisie à la question précédente.
3. Un palindrome est un mot qui peut être lu de gauche à droite ou de droite à gauche (ex : radar, kayak).
   — Déterminer si la chaîne de caractère est un palindrome en la parcourant avec le formalisme tableau (utilisation d'indices pour parcourir la chaîne).
   — Recommencez en utilisant deux pointeurs qui parcourent la chaîne de caractères.