Université de Valenciennes et du Hainaut-Cambrésis
ENSIAME **2017-2018**

Fiche d'Exercices Pratiques 1 : Pointeurs

Exercice 0 : Rappels

1. Le langage de programmation utilisé en TP est le 'C'.
2. Se connecter à l'environnement Linux si vous utilisez une machine de la salle TP, ou se connecter à votre PC.
3. Créer un répertoire `TP_APRI` sur votre espace mémoire, ainsi qu'un répertoire `TP2` pour y stocker les fichiers de cette séance.
4. Si vous êtes sur Linux :
   - Vous utiliserez `gedit` pour l'édition de vos fichiers. Ouvrez une fenêtre de console et taper `gedit &`
   - La compilation se fait **en ligne de commandes** dans une console. Si votre fichier s'appelle `tp2_ex1.c`, vous utiliserez la commande : `gcc -o tp2_ex1 tp2_ex1.c`
   - Si la compilation se termine sans erreur, vous pourrez tester votre programme en tapant la commande : `./tp2_ex1`

Exercice 1 :
Recopier, compiler, tester et vérifier que vous comprenez le code suivant.

```
#include <stdio.h>

int main()
{
    int a, b;
    int *p;

    a = 10;
    p = &a;
    b = *p;
    printf("Pointeur 'p' contient l'adresse de la variable 'a'\n");
    printf("Adresses: &a=%X, &b=%X, p=%X\n", &a, &b, p);
    printf("Valeurs: a=%d, b=%d, *p=%d\n\n", a, b, *p);

    a = 33;
    b = 20;
    printf("Pointeur 'p' contient l'adresse de la variable 'a'\n");
    printf("Adresses: &a=%X, &b=%X, p=%X\n", &a, &b, p);
    printf("Valeurs: a=%d, b=%d, *p=%d\n\n", a, b, *p);

    p = &b;
    printf("Pointeur 'p' contient l'adresse de la variable 'b'\n");
    printf("Adresses: &a=%X, &b=%X, p=%X\n", &a, &b, p);
    printf("Valeurs: a=%d, b=%d, *p=%d\n\n", a, b, *p);

  return 0;
}
```

Exercice 2 :

1. Écrire un sous-programme de saisie de deux valeurs réelles (passage par **adresse**).
2. Écrire un sous-programme qui permute le contenu de deux variables réelles (passage par **adresse**).
3. Tester ces deux sous-programmes avec un programme.

Exercice 3 :
Dans tout l'exercice on manipulera des pointeurs sur caractères pour utiliser des chaînes de caractères.

1. Écrire un sous-programme qui affiche un à un les caractères contenus dans une chaîne de caractères `ch` passée en paramètre.
2. Écrire un sous-programme qui calcule la longueur d'une chaîne de caractères.
3. Écrire un sous-programme qui recopie une chaîne `ch1` donnée à l'envers dans une autre chaîne `ch2`.
4. Tester ces sous-programmes avec un programme.

Exercice 4 :
Dans cet exercice on utilisera systématiquement un passage des paramètres par **adresse** (pointeurs).

Vous candidatez pour un poste d'informaticien au sein d'une société spécialisée dans la gestion de livres, et vous êtes retenu pour un entretien. Lors de celui-ci, le DRH vous propose une évaluation de vos compétences en algorithmique/langage 'C' à travers une réalisation (très) simplifiée d'un système de gestion des livres de la société. Vous disposez des informations suivantes sur le système : un **Livre** est représenté par un titre (chaîne de caractères), un identifiant (entier), un thème (chaîne de caractères) et une liste d'auteurs (tableau). Un **Auteur** est connu par ses nom et prénom (chaînes de caractères). Le système de gestion des livres, appelé **Biblio**, est défini par le nombre de livres présents et une liste (un tableau) de livres.

1. Définir les types nécessaires pour représenter le système de gestion de livres.
2. Écrire des sous-programmes de saisie d'un Auteur, d'un Livre, d'une Biblio.
3. Proposer un sous-programme qui dit si un Livre connu par son identifiant est dans une Biblio.
4. Tester ces premiers sous-programmes.
5. Proposer un sous-programme qui affiche les identifiants des livres d'un Auteur connu par son nom et son prénom (vous utiliserez la fonction `strcmp` définie dans la librairie `string.h` pour comparer les chaînes de caractères).
6. Écrire un sous-programme qui cherche et affiche l'identifiant du livre ayant le plus d'auteurs (un seul affichage en cas d'égalité).
7. Vérifier que ces sous-programmes fonctionnent en modifiant le programme.

Exercice 5 :
Dans cet exercice on utilisera systématiquement la notation avec une * pour accéder aux éléments du tableau et pas les [ ].

1. Définir un type tableau appelé `Tab` pouvant contenir au plus 50 entiers.
2. Écrire un sous-programme de saisie d'un tableau `t` de type `Tab` contenant `n` valeurs.
3. Écrire une fonction de recherche du maximum contenu dans un tableau `t` de type `Tab` ayant `n` valeurs.
4. Écrire un sous-programme qui dit si la somme des valeurs contenues dans les cases d'indices pairs est égale à celle des cases d'indices impairs, pour un tableau `t` de type `Tab` ayant `n` valeurs.
5. Tester les sous-programmes avec un programme principal.