Série 5 :
Programmation *C* - Pointeurs 1

Buts

Le but de cette série d'exercices est de vous permettre de pratiquer les bases concernant les pointeurs.

La série de cette semaine étant assez courte., profitez-en pour terminer les séries les plus longues des semaines précédentes.

Rappel

Avez-vous pris connaissance des **conseils relatifs à ces séries d'exercices** ?

Exercice 1 : Passage par référence (pointeurs, niveau 1)

NOTE : il faut faire cet exercice **sans** recopier le cours, mais par vous-même : quel doit être le prototype de `echange` ? comment procéder ? etc. Sinon cela n'a aucun intérêt !

Écrivez le programme `echange.c` dans lequel vous devez définir (prototype + définition) une fonction `echange` qui :

- accepte deux arguments de type `int*`.
- échange les valeurs de ces deux arguments.

Vous essayerez votre fonction avec le code `main` suivant :

```
int main(void)
{
    int i = 10;
    int j = 55;

    printf("Avant : i=%d et j=%d\n", i, j);
    echange(&i, &j);
    printf("Après : i=%d et j=%d\n", i, j);
    return 0;
}
```

Si votre fonction est correcte, le programme affichera :

```
Avant : i=10 et j=55
Après : i=55 et j=10
```

Exercice 2 : Pointeurs (niveau 1)

Un petit exercice simple pour manipuler les pointeurs comme sélecteurs d'objets.

Créez un nouveau fichier appelé `selecteur.c`.

Dans le `main`, créez trois variables `valeur1`, `valeur2` et `valeur3`, de type `double` que vous initialisez à des valeurs de votre choix.

Déclarez un pointeur `choix`, pointeur sur un type `double`.

Demandez ensuite un nombre entre 1 et 3 à l'utilisateur (avec une fonction `demander_nombre` par exemple) et, en selon son choix, faites pointer `choix` sur `valeur1`, `valeur2` ou `valeur3`.

Pour finir affichez «`Vous avez choisi` » et la valeur choisie **en utilisant le pointeur `choix`**.

Remarque: il est évident que le but de cet exercice n'est que de manipuler les pointeurs. Hors de ces contraintes, la bonne solution pour faire la même chose serait bien sûr d'utiliser un tableau.

Une meilleure illustration, mais plus complexe, de l'utilisation des pointeurs dans le choix d'objets est donnée dans l'exercice 6.

Exercice 3 : Explorer la mémoire (niveau 1)

Le but de cet exercice est de développer un « ausculteur » de mémoire rudimentaire : une fonction qui à partir d'une adresse et d'une taille affiche le contenu de chaque octet

Nous allons pour cela décomposer notre programme en plusieurs fonctions outils.

3.1 Affichage binaire

Le type approprié pour représenter un octet sans se préoccuper du bit de signe n'est pas « `char` », dont l'interprétation du signe n'est pas définie (« implementation defined »), mais bien « `unsigned char` ».

Commencez par définir le type « `octet` » comme « `unsigned char` ».

Ensuite, il n'exite pas de format à `printf` pour afficher en binaire.
Ecrivez une fonction « `afficher_binaire` » qui prend un `octet` et l'affiche en binaire. (A vous d'imaginer votre algorithme pour le faire (rappel cours ICC ??) ; il y a plusieurs solution possibles, plus ou moins avancées techniquement).

NOTE : pour afficher un seul caractère, « `printf("%c", a);` » peut s'écrire de façon plus efficace et plus compacte par : « `putchar(a);` ».

3.2 Affichage du i-ème octet

Ecrivez ensuite une fonction `affiche` qui prend un `size_t` et un octet et affiche :

- le `size_t` ; on pourra par exemple utiliser le format `printf` "%02zu" pour cela ;
- la valeur de l'octet en binaire ;
- la valeur de l'octet en entier positif ; on pourra par exemple utiliser le format `printf` "%3u" pour cela ;
- si la valeur de l'octet est comprise entre 32 et 126 (inclus) ; l'afficher en tant que caractère.

On pourrait par exemple avoir le résultat suivant :

```
00 : 01010000  80 'P'
```

3.3 Affichage du contenu de la mémoire

Ecrivez une fonction `dump_mem` prenant un pointeur sur un octet et une taille *N* (de type `size_t`) et qui :

- affiche l'adresse de départ ; pour afficher une adresse, on utilise le format `"%p"` de `printf` ;
- le contenu de tous les *N* octets à partir de l'adresse donnée.

On pourrait par exemple avoir le résultat suivant :

```
A partir de 0x7fffb7ed88ac :
00 : 01010000  80 ('P')
01 : 00000000   0
02 : 00000000   0
03 : 00000000   0
```

Pour afficher les adresses suivant un pointeur donné, utilisez simplement la syntaxe de tableau (nous reviendrons sur ce point plus tard dans le cours) : si `ptr` est votre pointeur, accédez à son contenu avec `ptr[0]`, puis au contenu juste après avec `ptr[1]`, etc. ; exactement comme si c'était un tableau.

3.4 Utilisation de l'outil et compréhention du contenu de la mémoire

Dans le `main`, définissez un `int` de valeur 80 (= 64 + 16), un autre valant -80, un `double` valant 0.5 et un autre valant 0.1 ; puis utilisez votre fonction `dump_mem` pour regarder comment ces valeurs sont représentées en mémoire.

Analysez/vérifiez les résultats obtenus (cf cours ICC ou alors peut être **ici : http://www.binaryconvert.com/convert_double.html**).

Dernière mise à jour le 10 mars 2016
Last modified: Thu Mar 10, 2016