

## R2.04 : programmation bas niveau en langage C

### TP1 – introduction au langage C et aux pointeurs

IUTLCO - Département Info – BUT Informatique

#### Préliminaires :

Prendre connaissances des documents sur Moodle – R2.04 :

- Fiche LAP – Langage C résumant la syntaxe du langage par rapport au LAP
- Cours avec explications et exemples de manipulation de pointeurs
- 2 fichiers « langage C dans VSC » pour rappel de l'installation et de l'utilisation du langage C dans Visual Studio Code.
- **Rappel** : en cas de problème avec VSC et le langage C sur votre système hôte, vous avez la machine virtuelle créée au semestre 1 qui est directement utilisable.

#### Contenu du TP :

Il consiste à manipuler :

- les variables
- les entrées/sorties avec *printf* et *scanf*
- les opérateurs de calcul et de comparaison
- les structures de base du langage : *if*, *else*, *switch*, *for*, *while*
- la définition et l'appel de fonctions
- le passage de paramètre par adresse dans des cas simples.

Les fonctions à écrire ont pour unique vocation de vous faire manipuler ces notions et ne seraient pas écrites de cette manière dans un programme C classique.

Le programme ne gère pas le fait que l'utilisateur saisisse autre chose que ce qui est demandé.

Ce TP est à déposer sur Moodle à l'issue de la séance.

#### Question 1 :

Écrire et tester la fonction *echanger(a,b)*, utilisant des entiers, en vous inspirant de la fonction init présentée en cours (diapo 47) et illustrant le passage de paramètre par adresse. Pour cela, suivre les étapes ci-dessous et utiliser la fiche LAP – Langage C ainsi que les exemples présentés en cours.

Créer un dossier pour vos TP de langage C, puis dans ce dossier, créer un fichier *tp1.c* débutant par l'inclusion des bibliothèques d'entrée/sortie et standard :

```
#include <stdio.h>
#include <stdlib.h>
```

Ajouter une fonction main et son contenu :

1. la fonction aura l'entête : *int main()*
2. définir 2 variables x et y de type entier
3. demander à l'utilisateur de saisir les valeurs de x et y avec des appels à *scanf*
4. afficher les valeurs saisies avec un appel à *printf*  
**Tester le programme (voir les fiches « langage C dans VSC »)**
5. écrire la fonction *echanger* avant la fonction *main*

6. dans la fonction *main*, après avoir affiché les valeurs x et y, faire un appel à la fonction *echanger* avec x et y passés en paramètres par adresse
7. réafficher les valeurs de x et de y après l'appel.

#### Tester le programme

#### Question 2 :

Compléter le même fichier tp1.c

Écrire et tester une fonction *trier(a,b,c)* permettant de réorganiser 3 entiers de manière à ce qu'après un appel à la fonction le premier indique le plus petit entier et le dernier le plus grand. Procéder de la même manière que dans la question 1 pour tester les valeurs de a, b et c avant et après l'appel à la fonction *trier*.

**Attention :** votre programme ne doit pas faire appel à une fonction de tri existante. Il doit être simple et utiliser un minimum de tests.

#### Question 3 :

Compléter le même fichier tp1.c

Modifier la fonction *main* de manière à gérer un menu qui permet de choisir de tester *echanger* ou alors *trier* ou alors de quitter. Le menu se présentera comme ci-dessous :

```
1. echanger
2. trier
0. quitter
? █
```

Le menu attend que l'utilisateur saisisse 1, 2 ou 0 et boucle tant que l'utilisateur ne quitte pas. Si l'utilisateur saisit un numéro qui n'existe pas dans le menu alors le message « Valeur non reconnue. » est affichée et le menu boucle.

La boucle du menu doit être gérée à l'aide d'un *while*.  
Le choix fait par l'utilisateur doit être géré par un *switch*.

#### Question 4 :

Compléter le même fichier tp1.c

1. Ajouter un menu 3 intitulé « puissance2 »
2. Écrire la fonction *fois2(a)* dont le paramètre est un pointeur vers un entier qui est multiplié par 2 à chaque appel de cette fonction. Exemple : si l'entier i vaut 5, alors le fait d'appeler *fois2* avec i passé par adresse va modifier i en 10.  
Tester la fonction *fois2* dans le menu 3.
3. Écrire la fonction *puissance2(n)* (le paramètre n est passé par valeur) qui calcule et retourne  $2^n$  en utilisant une boucle *for* qui appelle n fois la fonction *fois2*, puis retourne le résultat.  
Tester cette fonction *puissance2* dans le menu 3 à la place de *fois2* : lire la valeur de l'exposant puis appeler *puissance2*. Le résultat de la fonction doit être récupéré et affiché.

#### Question 5 :

Compléter le même fichier tp1.c

Dans la question précédente, le résultat est retourné à la fin de la fonction *puissance2*.

Le but de cette dernière question est de passer la puissance à calculer en paramètre par adresse au lieu de la retourner.

1. Ajouter un menu 4 intitulé « puissance2\_par\_adresse »
2. Écrire une fonction *puissance2\_par\_adresse(x,n)* en reprenant la fonction *puissance2* mais qui utilise le paramètre *x* passé par adresse pour calculer la puissance. C'est donc *x* qui sera passé en paramètre de la fonction *fois2* dans la boucle *for*. Cette fonction ne retournera donc aucun résultat.
3. Tester la fonction dans le menu 4.

#### Question 6 :

Modifier la fonction *trier* pour que celle-ci fasse appel à la fonction *echanger* au lieu de faire les échanges manuellement. Tester son fonctionnement.

#### Question 7 :

Compléter le même fichier tp1.c

1. Ajouter un menu 5 intitulé « pgcd-ppcm »
2. Lire 2 nombres entiers *x* et *y* non nuls (si l'utilisateur entre un nombre nul, le relire)
3. Créer une fonction *pgcd\_ppcm(x,y,pgcd,ppcm)* qui prend 4 paramètres, 2 nombres entiers non nuls et le pgcd et le ppcm de ces 2 nombres qui seront calculés :
  1. le pgcd sera calculé à l'aide d'une boucle *while* en suivant l'algorithme d'Euclide : [https://fr.wikipedia.org/wiki/Algorithme\\_d%27Euclide](https://fr.wikipedia.org/wiki/Algorithme_d%27Euclide)
  2. le ppcm sera calculé à l'aide du pgcd : [https://fr.wikipedia.org/wiki/Plus\\_petit\\_commun\\_multiple](https://fr.wikipedia.org/wiki/Plus_petit_commun_multiple)
4. Tester la fonction

**Le programme terminé :** Votre programme doit fonctionner de la même manière.

```
1. echanger
2. trier
3. puissance2
4. puissance2_par_adresse
5. pgcd-ppcm
0. quitter
Votre choix ? 2

Test de trier
Saisir 3 entiers : 14 9 42
avant trier : x=14 y=9 z=42
apres trier : x=9 y=14 z=42

1. echanger
2. trier
3. puissance2
4. puissance2_par_adresse
5. pgcd-ppcm
0. quitter
Votre choix ? 4

Test de puissance2_par_adresse
Saisir un entier : 5
Calcul de 2^5
avant puissance2_par_adresse : x=0
apres puissance2_par_adresse : x=64

1. echanger
2. trier
3. puissance2
4. puissance2_par_adresse
5. pgcd-ppcm
0. quitter
Votre choix ? 5
```

```
Test de pgcd-ppcm
Saisir 2 entiers strictement positifs : 0 142

Les 2 entiers doivent etre strictement positifs, nouvelle saisie : 384 142
Le pgcd de 384 et 142 est : 2
Le ppcm de 384 et 142 est : 27264

1. echanger
2. trier
3. puissance2
4. puissance2_par_adresse
5. pgcd-ppcm
0. quitter
Votre choix ? 7

Valeur non reconnue.

1. echanger
2. trier
3. puissance2
4. puissance2_par_adresse
5. pgcd-ppcm
0. quitter
Votre choix ? 0

Fin du programme.
```