Feuille de TD1 : Utilisation des pointeurs

Tous les exercices devront se faire dans le « home » de linux dans un dossier appelé « systeme ». (mkdir systeme puis cd systeme).

Rappel du cours :

if (test) **then** {…} **else** {…}
while(test){ … }
for(i=0;i<n;i++){ … }
sleep(n); //dort n secondes
printf(‘’ … %d \n ‘’, entier);
exit(1); // fini le processus
scanf(‘’%d ‘’, &entier);

&*x* : retourne l’adresse de la variable *x.*
**x* : accède à la variable stockée à l’adresse.

```
typedef struct{
        int a;
        int b;
}nom_de_la_structure;
```

Exercice 0 : Premier code c …

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

int main(int argc, char *argv[] ){
 //argc : nombre d’arguments, argv : le tableau contenant la valeur des arguments
 int a = 1 ;
 printf(‘’ le chiffre est %d \n ‘’, a);
 char clavier[100];
 printf(‘’ Saisissez le texte : ‘’);
 scanf(‘’%s’’, clavier);
 printf(‘’%s \n’’,clavier) ;
 int i;
 for(i=0;i<argc;i++){
  printf(‘’%s \n’’, argv[i]);
 }
}
```

Compilation : gcc nom_fichier.c –o nom_executable
Exécution : ./nom_executable test_affichage1 test_affichage2

Exercice 1 : Pointeurs simples

Créer trois fonctions « plus », « moins » et « fois ». Chacune des trois fonctions prend 3 paramètres de type pointeur sur des *entiers* (int*) et effectue l'opération entre les deux premiers paramètres pour stocker le résultat dans le troisième. Dans le *main,* définir 3 variables de type entier *x*, *y* et *z* puis tester vos 3 fonctions en affichant le résultat entre chaque utilisation d'une fonction.

Exercice 2 : Pointeurs avec structure

Définir une structure « coordonnees » contenant 3 entiers (x,y,z). La fonction *main* déclare une variable *ici* de type « coordonnees ». Elle envoie l'adresse de cette structure à une fonction « initialise » qui initialise respectivement les valeurs x, y et z à 10, 5 et 2. Ensuite, la fonction main envoie l'adresse de cette même structure successivement aux 3 fonctions suivantes :

- affichage : affiche la valeur de x, y et z,
- multiplication : multiplie x et y puis stocke le contenu dans z,
- addition : additionne y et z puis stocke le contenu dans z.

(La fonction affichage peut être utilisée plusieurs fois afin de voir l'évolution des opérations).

Exercice 3 : Pointeurs plus

Reprendre le code précédent mais cette fois la fonction d'affichage sera appelée dans les fonctions multiplication et addition.

Rappel d'utilisation des tableaux

```
char * pointeur = malloc(20 * sizeof(char)); //Allocation de 20 caractères
if(pointeur == NULL) {
  printf("L'allocation n'a pu être réalisée\n");
} else {
  printf("L'allocation a été un succès\n");
  free(pointeur); //Libération des 20 octets précédemment alloués
  pointeur = NULL; // Invalidation du pointeur
}
```

```
char * tmp = realloc(tampon, nouvelle_taille);
if (tmp == NULL) {
   free(tampon);
   rapporterErreur();
} else
   tampon = tmp;
```

Dans les exercices qui suivent n'oubliez pas d'afficher le tableau (ou matrice) entre chaque changement.

Exercice 4 : Création de tableau dynamique

Proposer un programme *c* qui demande de saisir une taille de tableau, puis alloue dynamiquement un tableau d'entiers de cette taille, le remplit avec des valeurs aléatoires comprises entre 0 et 100 et l'affiche.

Exercice 5 : Modification de la taille du tableau

Reprendre le code précédent et en plus on changera la taille du tableau (en gardant les données initiales) en fonction d'une nouvelle taille saisie par l'utilisateur. On affichera le nouveau tableau.

Exercice 6 : Création d'une matrice dynamique

Proposer un programme *c* qui alloue dynamiquement une matrice (int ** matrice) d'entiers, la remplit avec des valeurs aléatoires et l'affiche. Le nombre de lignes de colonnes étant saisi au clavier au préalable.

Exercice 7 : Modification de matrice dynamique

Reprendre le code précédent et en plus on changera la taille de la matrice en fonction de deux nouveaux nombres de lignes et colonnes saisis au clavier. La matrice doit garder les anciennes valeurs. On affichera bien-sûr la nouvelle matrice.

Exercice 8 : Tableau de chaines de caractères

Proposer un programme *c* qui saisi un nombre n de chaines. Alloue dynamiquement un tableau de n chaines de caractères et le remplit avec des valeurs saisies dans le terminal. Le programme doit afficher à la fin les chaines du tableau.