Système TD 1

EXERCICE 0 : premier code C

```c
int main() {
    // argc : nombre d'arguments
    // argv : le tableau contenant les valeurs des arguments
    int a = 1;
    printf("le chiffre est %d\n", a);
    char clavier[100];
    printf("Saisissez le texte :\n");
    scanf("%s\n", clavier);

    for (int i = 0;  i < argc; i++) {
        printf("%s\n", argv[i]);
    }
}
```

Compilation : gcc nom_du_fichier.c -o nom_executable
Exécution : ./nom_executable

EXERCICE 1 : pointeurs simples

Créer 3 fonctions “*plus*”, “*moins*” et “*fois*”.
Chacune des trois fonctions prend 3 paramètres de type **pointeur** d’entier (*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* (x=50, y=20, z=0).
Puis tester les 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 variable à une fonction "*initialiser*" qui initialise respectivement les valeurs *x*, *y* et *z* à 1, 5 et 2.
Ensuite, la fonction *main()* envoie l'adresse de la variable *ici* aux 3 fonctions suivantes :

- afficher : affiche la valeur de *x*, *y* et *z* ;
- multiplier : multiplie *x* et *y* puis stocke le résultat dans *z* ;
- additionner : additionne *x* et *y* puis stocke le résultat dans *z*.

La fonction "*afficher*" peut être utilisée plusieurs fois afin de voir l'évolution des opérations.

EXERCICE 3 : pointeurs plus

Reprendre l'exercice précédent, mais cette fois la fonction "*afficher*" sera appelée dans les fonctions de multiplication et d'addition.

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 (voir rappel en bas de la feuille), le remplit avec des valeurs aléatoires comprises entre 0 et 100 et l'affiche.

EXERCICE 5 : modification de tableau dynamique

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 (voir rappel).
On affichera le nouveau tableau.

EXERCICE 6 : création de matrice dynamique

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

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 de colonnes saisis au clavier. La matrice doit garder les anciennes valeurs.
On affichera la nouvelle matrice.

EXERCICE 8 : tableau de chaînes de caractères

Proposer un programme C qui saisit un nombre *N* de chaînes. Puis alloue dynamique un tableau de *N* chaînes de caractères et le remplit avec des valeurs saisies dans le terminal.
Le programme doit afficher, à la fin, les chaînes du tableau.

RAPPEL

Création d'un tableau dynamique.

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

Modification d'un tableau.

```c
char * tmp = realloc(tableau, nouvelleTaille * sizeof(char));
if (tmp == NULL) {
    free(tableau);
    printf("La réallocation n'a pu être réalisée\n");
} else {
    tableau = tmp;
}
```