# Série de (TD et TP) N°1

[Les tableaux et les pointeurs]

**Exercice 1 :** Soit un tableau T de type entier. Définir le résultat des opérations suivantes :

1. int tab[4] = {10, 23,505,8};
2. int tableau[4] = {10, 23};
3. *printf*("%-", tableau[0]);
4. *printf*("%-", *tableau);
5. *printf*("%-", tableau);
6. *printf*("%-", *(tableau+1));
7. *printf*("%-", tableau+1);

**Exercice 2 :** Combien d'octets réservés/occupés par :

1. Une variable de type **char**
2. Une variable de type **int**
3. Une variable de type **long**
4. Une variable de type **double**
5. Une structure composée de deux variables de type **int**
6. Un tableau **int T[100]**

**Exercice 3 :** Un programme C permet de demander à l'utilisateur de saisir son âge, puis le programme affiche la valeur saisie.

- Ecrire le programme en utilisant une variable statique
- Récrire le programme en utilisant une variable dynamique

**Exercice 4 :**

1. Écrire deux fonctions permettant d'afficher une variable de type entier par :
   - Passage de valeur
   - Passage d'adresse
2. Écrire deux fonctions permettant d'initialiser une variable de type entier par :
   - Passage de valeur
   - Passage d'adresse

**Exercice 5 :** Soit un tableau d'entiers de taille 100, dont les premiers éléments sont 10, 23, 505, 8, et le reste sont initialisés à 0. Ecrire une fonction pour afficher les valeurs du tableau comme suit :

1. Une fonction qui reçoit le tableau (passage par valeur)
2. Une fonction qui reçoit le tableau (passage par adresse)

**Exercice 6 :** Soient deux tableaux des entiers :

1. Créez une fonction *sommeTableau* qui renvoie la somme des valeurs contenues dans le tableau
2. Créez une fonction *moyenneTableau* qui calcule et renvoie la moyenne des valeurs.
3. Créez une fonction *copierTableau* qui prend en paramètre deux tableaux.

**Exercice 7 :** Ecrire un programme qui permet de demander à l'utilisateur de saisir le nombre de ses amis, puis il demande à l'utilisateur de remplir un tableau contenant l'âge de ses amis.

**Exercice 8 :** Écrire un programme qui lit un entier X et un tableau A du type int au clavier et élimine toutes les occurrences de X dans A en tassant les éléments restants. Le programme utilisera les pointeurs P1 et P2 pour parcourir le tableau.

**Exercice 9 :** Écrire un programme qui range les éléments d'un tableau A du type int dans l'ordre inverse.

**Exercice 10 :** Soient deux tableaux d'entiers. Ecrire un programme en C qui permet de tester l'égalité entre les deux tableaux : il rend VRAI si les composants des deux tableaux correspondent position par position, et FAUX sinon.

# Corrigé de Série de (TD et TP) N°1

[Les tableaux et les pointeurs]

**Exercice 1 :** Résultats d'exécution des instructions :

1. 10 | 23 | 505 | 8
2. 10 |23 |0 | 0
3. 10
4. On voit l'adresse où se trouve ***tableau[0]***
5. 23
6. On voit l'adresse où se trouve ***tableau[1]***

**Exercice 2 :** Combien d'octets réservés/occupés par :

1. Une variable de type **char : 1 octet**
2. Une variable de type **int : 4 octets**
3. Une variable de type **long : 4 octets**
4. Une variable de type **double : 8 octets**
5. Une structure composée de deux variables de type **int : 8 octets**
6. Un tableau **int T[100] : 400 octets**

```c
printf("char : %d octets\n", sizeof(char));
printf("int : %d octets\n", sizeof(int));
printf("long : %d octets\n", sizeof(long));
printf("double : %d octets\n", sizeof(double));
```

## Exercice 3 :

```c
int main(int argc, char *argv[])
{
    int maVariable = 0; // Allocation de la mémoire (automatique)

    // Utilisation de la mémoire
    printf("Quel age avez-vous ? ");
    scanf("%d", &maVariable);
    printf("Vous avez %d ans\n", maVariable);

    return 0;
} // Libération de la mémoire (automatique à la fin de la fonction)
```

```c
int main(int argc, char *argv[])
{
    int* memoireAllouee = NULL;

    memoireAllouee = malloc(sizeof(int)); // Allocation de la mémoire
    if (memoireAllouee == NULL)
    {
        exit(0);
    }

    // Utilisation de la mémoire
    printf("Quel age avez-vous ? ");
    scanf("%d", memoireAllouee);
    printf("Vous avez %d ans\n", *memoireAllouee);

    free(memoireAllouee); // Libération de mémoire

    return 0;
}
```

**Exercice 4 :**

```c
#include <stdio.h>
//Question 1
void afficherParValeur(int val);
void afficherParAdresse(int* val);
//Question2
int initParValeur();
void initParAdresse(int* val);

int main()
{
    int val;
    //pour initialiser par passage de valeur
    val = initParValeur(val);
    //pour afficher par passage de valeur
    afficherParValeur(val);
    //pour afficher par passage par adresse
    afficherParAdresse(&val);

    //pour initialiser par passage de valeur
    initParAdresse(&val);
    afficherParValeur(val);

    return 0;
}

void afficherParValeur(int val) {
    printf("%d\n", val);
}

void afficherParAdresse(int* val) {
    printf("%d\n", *val);
}

int initParValeur() {
    int val;
    printf("Saisir une valeur:");
    scanf("%d\n", &val);
    return val;
}

void initParAdresse(int* val) {
    scanf("Saisir une valeur:%d\n", val);
}
```

**Exercice 5 :**

```c
#include <stdio.h>

void afficher(const int* tab, int taille);

int main()
{
    int tableau[100] = {10,2,4,3};
    //passage par valeur
    afficher(tableau, 100);
    //passage par adresse
    //afficher(tableau, 4);
    printf("%d\n", *tableau);
    return 0;
}
//pour les tableau on utilise deux ecritures
//int* tab ou int tab[]
//déclarer le tableau comme const si vous vouler
void afficher(const int* tab, int taille) {
    printf("adresse du tableau:%p\n", tab);
    printf("Affichage des valeurs du tableau:\n");
    int i;
    for (int i = 0; i < taille; i++) {
        printf("%d\n", *(tab+i));//c'est l'équivalent de
        //printf("%d\n", tab[i]);
    }
    //pour tester si la valeur de tab[0] change
    //parce que notre tableau est déclaré comme const
    //cette instruction va nous donner une erreur
    tab[0] = 0;
}
```

**Exercice 6 :** soient deux tableaux des entiers :

```c
int sommeTableau(int tableau[], int tailleTableau);
```

```c
double moyenneTableau(int tableau[], int tailleTableau);
```

```c
void copie(int tableauOriginal[], int tableauCopie[], int tailleTableau);
```

**Exercice 7 :** Ecrire un programme qui permet de demander à l'utilisateur de saisir le nombre de ses amis, puis il demande à l'utilisateur de remplir un tableau contenant l'âge de ses amis.

1. Demander à l'utilisateur combien il a d'amis ;
2. Créer un tableau de entiers ayant une taille égale à son nombre d'amis (malloc) ;
3. Afficher l'âge des amis pour montrer qu'on a bien mémorisé tout cela ;
4. A la fin, puisqu'on n'a plus besoin du tableau contenant l'âge des amis, le libérer avec la fonction free ;

```c
int main(int argc, char *argv[])
{
    int nombreDAmis = 0, i = 0;
    int* ageAmis = NULL; // Ce pointeur va servir de tableau après l'appel du malloc

    // On demande le nombre d'amis à l'utilisateur
    printf("Combien d'amis avez-vous ? ");
    scanf("%d", &nombreDAmis);

    if (nombreDAmis > 0) // Il faut qu'il ait au moins un ami (je le plains un peu sinon :p)
    {
        ageAmis = malloc(nombreDAmis * sizeof(int)); // On alloue de la mémoire pour le tableau
        if (ageAmis == NULL) // On vérifie si l'allocation a marché ou non
        {
            exit(0); // On arrête tout
        }

        // On demande l'âge des amis un à un
        for (i = 0 ; i < nombreDAmis ; i++)
        {
            printf("Quel age a l'ami numero %d ? ", i + 1);
            scanf("%d", &ageAmis[i]);
        }

        // On affiche les âges stockés un à un
        printf("\n\nVos amis ont les ages suivants :\n");
        for (i = 0 ; i < nombreDAmis ; i++)
        {
            printf("%d ans\n", ageAmis[i]);
        }

        // On libère la mémoire allouée avec malloc, on n'en a plus besoin
        free(ageAmis);
    }

    return 0;
}
```