

## TD1 : Allocation Dynamique de Mémoire & Enregistrements

### Exercice N° 1 :

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

int main() {
    int n, i;
    float *notes;
    float somme, moyenne, min, max;

    printf("Nombre d'etudiants : ");
    scanf("%d", &n);

    while (n <= 0) {
        printf("Nombre invalide.\n");
        scanf("%d", &n);
    }

    // Allocation de la mémoire
    notes = malloc(n * sizeof(float));

    if (notes == NULL) {
        printf("Erreur d'allocation memoire.\n");
        return 1;
    }

    for (i = 0; i < n; i++) {
        printf("Note de l'etudiant %d : ", i + 1);
        scanf("%f", &notes[i]);
    }

    somme = 0;
    min = notes[0];
    max = notes[0];

    for (i = 0; i < n; i++) {
        somme += notes[i];

        if (notes[i] > max)
            max = notes[i];
    }
}
```

```

        if (notes[i] < min)
            min = notes[i];
    }

    moyenne = somme / n;

    printf("\nMoyenne : %.2f\n", moyenne);
    printf("Note maximale : %.2f\n", max);
    printf("Note minimale : %.2f\n", min);

    free(notes);

    return 0;
}

```

## **Exercice N° 2 :**

```

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

void saisirVentes(int *ventes, int n) {
    int i;

    for (i = 0; i < n; i++) {
        printf("Ventes du jour %d : ", i + 1);
        scanf("%d", &ventes[i]);
    }
}

void afficherVentes(int *ventes, int n) {
    int i;

    printf("\n--- Ventes ---\n");

    for (i = 0; i < n; i++) {
        printf("Jour %d : %d\n", i + 1, ventes[i]);
    }
}

int totalVentes(int *ventes, int n) {
    int i;
    int total = 0;

    for (i = 0; i < n; i++) {
        total += ventes[i];
    }

    return total;
}

```

```

}

int jourMeilleureVente(int *ventes, int n) {
    int i;
    int indiceMax = 0;

    for (i = 1; i < n; i++) {
        if (ventes[i] > ventes[indiceMax]) {
            indiceMax = i;
        }
    }

    return indiceMax + 1;
}

int main() {
    int n;
    int *ventes;

    printf("Nombre de jours : ");
    scanf("%d", &n);

    if (n <= 0) {
        printf("Nombre invalide.\n");
        return 1;
    }

    ventes = malloc(n * sizeof(int));

    if (ventes == NULL) {
        printf("Erreur d'allocation memoire.\n");
        return 1;
    }

    saisirVentes(ventes, n);

    afficherVentes(ventes, n);

    printf("\nTotal des ventes : %d\n",
        totalVentes(ventes, n));

    printf("Jour de la meilleure vente : %d\n",
        jourMeilleureVente(ventes, n));

    free(ventes);

    return 0;
}

```

### Exercice N° 3 :

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

// Déclaration de la structure Etudiant
typedef struct {
    char nom[30];
    int age;
    float note;
} Etudiant;

// Fonction qui lit et retourne un pointeur sur un étudiant
Etudiant* lireEtudiant() {
    Etudiant* e = (Etudiant*)malloc(sizeof(Etudiant)); // allocation
    dynamique

    if (e == NULL) { // Vérification d'allocation
        printf("Erreur d'allocation mémoire !\n");
        exit(1);
    }

    printf("Entrez le nom de l'étudiant : ");
    scanf("%s", e->nom);

    printf("Entrez l'âge de l'étudiant : ");
    scanf("%d", &e->age);

    printf("Entrez la note de l'étudiant : ");
    scanf("%f", &e->note);

    return e; // renvoie un pointeur sur la structure allouée
}

// Fonction d'affichage d'un étudiant
void afficherEtudiant(Etudiant etu) {
    printf("Nom : %s\t Âge : %d\t Note : %.2f\n", etu.nom, etu.age, etu.note);
}

// Fonction qui calcule la moyenne d'un tableau d'étudiants
float moyenne(Etudiant *et, int n) {
    float somme = 0;
    for (int i = 0; i < n; i++) {
        somme += et[i].note;
    }
    return somme / n;
}
```

```

// Fonction principale
int main() {
    int n;

    printf("Entrez le nombre d'étudiants : ");
    scanf("%d", &n);

    // Allocation dynamique d'un tableau de n étudiants
    Etudiant *tab = (Etudiant*)malloc(n * sizeof(Etudiant));

    if (tab == NULL) {
        printf("Erreur d'allocation mémoire !\n");
        return 1;
    }

    // Lecture des informations des étudiants
    for (int i = 0; i < n; i++) {
        printf("\n--- Étudiant %d ---\n", i + 1);
        Etudiant *temp = lireEtudiant(); // retourne un pointeur sur un
étudiant
        tab[i] = *temp;                  // copie le contenu dans le tableau
        free(temp);                      // libère la mémoire temporaire
    }

    // Affichage de la liste des étudiants
    printf("\n=== Liste des étudiants ===\n");
    for (int i = 0; i < n; i++) {
        afficherEtudiant(tab[i]);
    }

    // Calcul et affichage de la moyenne
    float moy = moyenne(tab, n);
    printf("\nMoyenne générale : %.2f\n", moy);

    // Libération de la mémoire
    free(tab);

    return 0;
}

```

#### **Exercice N° 4 :**

```

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

// Définition des structures

```

```

typedef struct {
    char ville[30];
    char rue[30];
    int codePostal;
} Adresse;

typedef struct {
    char nom[30];
    Adresse adr; // champ de type Adresse
} Employe;

// Fonction pour lire un employé
Employe* lireEmploye() {
    Employe *e = (Employe*)malloc(sizeof(Employe));

    if (e == NULL) {
        printf("Erreur d'allocation mémoire !\n");
        exit(1);
    }

    printf("Entrez le nom de l'employé : ");
    scanf("%s", e->nom);

    printf("Entrez la ville : ");
    scanf("%s", e->adr.ville);

    printf("Entrez la rue : ");
    scanf("%s", e->adr.rue);

    printf("Entrez le code postal : ");
    scanf("%d", &e->adr.codePostal);

    return e;
}

// Fonction pour afficher un employé
void afficherEmploye(Employe e) {
    printf("Nom : %s\n", e.nom);
    printf("Ville : %s\n", e.adr.ville);
    printf("Rue : %s\n", e.adr.rue);
    printf("Code postal : %d\n", e.adr.codePostal);
    printf("-----\n");
}

// Fonction principale
int main() {
    int n;
    printf("Entrez le nombre d'employés : ");

```

```

scanf("%d", &n);

Employe *tab = (Employe*)malloc(n * sizeof(Employe));
if (tab == NULL) {
    printf("Erreur d'allocation mémoire !\n");
    return 1;
}

// Lecture des employés
for (int i = 0; i < n; i++) {
    printf("\n--- Employé %d ---\n", i + 1);
    Employe *temp = lireEmploye();
    tab[i] = *temp; // copie du contenu
    free(temp);
}

// Affichage des employés
printf("\n=== Liste des employés ===\n");
for (int i = 0; i < n; i++) {
    afficherEmploye(tab[i]);
}

free(tab);
return 0;
}

```