

INTRODUCTION AUX TYPES STRUCTURÉS

Exercice 1 : Le problème sans structure

On souhaite gérer les informations d'un étudiant :

- son prénom,
- son âge,
- sa moyenne.

Question 1 Proposer une liste de variables pour stocker ces informations.

Question 2 Écrire un code qui définit puis affiche les informations d'une étudiante nommée « Alice », 20 ans, moyenne 14.2.

Question 3 Comment stocker les informations de 3 étudiants différents ?

Question 4 Pourquoi cette approche devient-elle source d'erreurs ?



Les types structurés en C

En C, on peut regrouper plusieurs variables dans une seule entité appelée **structure**, définie avec le mot-clé **struct**.

Dans notre cas, nous allons créer un type structuré qui contiendra différentes informations concernant une même entité.

La définition d'un type structuré se fait généralement avec le mot-clé **typedef** pour simplifier son utilisation ultérieure. La syntaxe générale est la suivante :

```
5   typedef struct {
        type1 champ1;
        type2 champ2;
        ...
        typeN champN;
    } NomDeLaStructure;
```

Prenons l'exemple d'un contact dans un carnet d'adresses. On peut définir une structure **Contact** qui regroupe plusieurs champs :

```
5   typedef struct {
        string nom;
        string prenom;
        string telephone;
        string email;
        int anneeNaissance;
    } T_Contact;
```

Exercice 2 : Créer une structure cohérente

On regroupe les données dans une seule entité : un **type structuré**.

Question 1 À l'aide de l'exemple précédent, écris la définition d'une structure **Etudiant** contenant les champs correspondants aux informations d'un étudiant (prénom, âge, moyenne).

Question 2 Écrire les déclarations suivantes :

- une variable **etudiant_1** représentant un étudiant,
- un tableau **classe** de 30 étudiants.

Question 3 Écrire les lignes de code permettant d'initialiser les informations de l'étudiant **etudiant_1** avec les valeurs suivantes : prénom « Bob », âge 22, moyenne 15.5.

Question 4 Écrire les lignes de code permettant de demander à l'utilisateur de saisir les informations des 30 étudiants du tableau **classe**.

Question 5 Écrire les lignes de code permettant d'afficher le prénom et la moyenne du meilleur étudiant de la classe.

Exercice 3 : Représenter une heure

On veut manipuler une heure sous la forme HH:MM:SS.

Question 1 Proposer un type structuré **T_Horaire** permettant de stocker les informations d'un horaire.

Question 2 Écrire une fonction **void afficher_heure(T_Horaire t)** qui affiche HH:MM:SS.

Question 3 Écrire **int en_secondes(T_Horaire t)** qui renvoie le nombre total de secondes.

Question 4 Créer deux horaires dans le **main** et afficher la différence en secondes.

Question 5 Ecrire la fonction inverse qui prend en paramètre un entier représentant un nombre de secondes et qui renvoie une structure **T_Horaire** correspondante.

Exercice 4 : Gérer un produit en stock

On veut garder les informations suivantes pour chaque produit :

- son nom (chaîne de caractères),
- son prix HT (nombre à virgule),
- son taux de TVA (nombre à virgule).

Question 1 Proposer un type structuré **T_Produit** pour stocker ces informations.

Question 2 Écrire une fonction qui renvoie le prix TTC d'un produit (la fonction ne prendra qu'un seul argument).

Exercice 5 : Gestion d'un vecteur 2D

On manipule des vecteurs dans un plan :

```
typedef struct{
    float x;
    float y;
} T_Vecteur;
```

Question 1 Écrire une fonction qui renvoie la norme $\sqrt{x^2 + y^2}$ d'un vecteur donné en paramètre.

Question 2 Écrire une fonction permettant de sommer deux vecteurs.

Question 3 Écrire une fonction permettant de donner l'opposé d'un vecteur.

Question 4 Dans le **main**, déclarer trois vecteurs, afficher leurs normes, puis afficher la norme de $a - b$.

Exercice 6 : Extension : struct imbriquées

Il est également possible de créer des structures imbriquées, c'est-à-dire d'utiliser une structure comme champ d'une autre structure. Par exemple, on peut définir une structure **Adresse** et l'utiliser dans une structure **Etudiant** :

```

typedef struct {
    string rue;
    int codePostal;
    string ville;
} T_Adresse;

typedef struct {
    string prenom;
    int age;
    float moyenne;
    T_Adresse adresse;
} T_EtudiantAdresse;
```

Question 1 Comment accéder à la ville d'un étudiant ?

Question 2 Quel est l'avantage de créer une structure imbriquée plutôt qu'ajouter 3 champs à **Etudiant** ?

Question 3 Proposer une fonction `int habite_ville(T_EtudiantAdresse e, const char* ville)` qui renvoie 1 si l'étudiant habite dans la ville donnée, 0 sinon.

Exercice 7 : Projet d'ensemble : bulletins d'une classe

On souhaite écrire un petit programme qui gère les **bulletins** d'une classe. Chaque élève a plusieurs notes dans différentes matières, avec un **coefficent** pour chaque matière.

On suppose que :

- tous les élèves ont les **mêmes matières**, dans le même ordre,
- il y a au maximum 30 élèves dans la classe,
- il y a par exemple 4 matières par élève,
- chaque élève peut avoir **plusieurs notes** par matière (par ex. contrôles, devoirs, etc.).

Question 1 Proposer des constantes et des types structurés permettant de modéliser ce problème.

Question 2 Écrire une fonction `float moyenne_eleve(T_Eleve e, T_Matiere matieres[])` qui renvoie la moyenne générale de l'élève, en tenant compte des coefficients de chaque matière.

Question 3 Écrire une fonction `float moyenne_matiere(T_Eleve classe[], int nbEleves, int indiceMatiere)` qui renvoie la moyenne de la classe pour une matière donnée (identifiée par son indice).

Question 4 On va maintenant écrire une partie de `main`. On dispose d'un tableau `T_Eleve classe[TAILLE_MAX_CLASSE];` et d'un tableau `T_Matiere matieres[NB_MATIERES];`. Écrire le code permettant :

1. de demander à l'utilisateur combien d'élèves sont dans la classe ;
2. de saisir les matières (nom et coefficient) une seule fois ;
3. de saisir pour chaque élève : son prénom, son âge, puis ses notes dans les matières ;
4. d'afficher, pour chaque élève, sa moyenne générale.

Question 5 À partir des fonctions précédentes, écrire le code (dans le `main` ou dans une fonction dédiée) qui :

- détermine l'élève ayant la meilleure moyenne générale et affiche son prénom et sa moyenne ;
- calcule la moyenne de la classe pour chaque matière ;
- affiche la matière dont la moyenne de classe est la plus faible (matière « à renforcer »).