HE VD IG

# Série 0x22

**Solution**

*Les structures de données*

INFO2-TIN — 3 mars 2026, 7efbe5

## Exercice 1 : Connaissances générales

**(a)** Qu'est-ce qu'une structure de données en langage C ?

A. Un type de données primitif ;
**B. Un type de données composés qui regroupe plusieurs éléments de données ;**
C. Une manière d'agencer des fichiers sur un disque dur ;
D. Un algorithme de tri.

**(a) B**

**(b)** Quelle est la syntaxe correcte pour déclarer une structure de données en langage C ?

A. `struct { ... }`
B. `typedef struct { ... }`
C. `struct name { ... }`
**D. `typedef struct name { ... } Name;`**
E. `typedef struct { ... } Name;`
F. `struct name { ... };`

**(b) D**

> On privilégie toujours la création d'un type avec `typedef` pour raccourcis la syntaxe d'utilisation et éviter l'utilisation de `struct` à chaque fois. De plus, il est recommandé de donner un nom à la structure pour faciliter le débogage et la lisibilité du code. Enfin on choisit si possible un nom de type en PascalCase pour différencier les types des variables, et s'approcher de la syntaxe des langages orientés objet (Python, C++, C#)...

**(c)** Comment connaître la taille en octets d'une structure de données en langage C, en admettant que la structure n'est pas un type incomplet ?

A. En utilisant l'opérateur `sizeof` sur une variable de ce type ;
B. En utilisant l'opérateur `sizeof` sur le type de la structure ;
**C. En utilisant l'opérateur `sizeof` sur le type de la structure ;**
D. En utilisant l'opérateur `strlen` ;
E. En utilisant l'arithmétique de pointeurs ;
F. On ne peut pas connaître la taille d'une structure de données.

**(c) C**

**(d)** Qu'est-ce qu'un membre flexible dans une structure de données en langage C ?

A. Un membre qui peut être modifié à l'exécution ;
B. Un membre qui peut être de n'importe quel type ;
**C. Un membre qui peut être un tableau de taille variable, défini à la fin de la structure ;**
D. Un membre qui peut être utilisé pour stocker des données de type `void *`.

**(d) C**

**(e)** Comment une variable de type `struct` est passée à une fonction ?

A. Passage par valeur (copie du contenu) ;
B. Passage par adresse (copie de l'adresse).

**(e)**

## Exercice 2 : Anniversaires

On réalise un programme permettant de traiter les anniversaires de plusieurs personnes. Chaque personne est caractérisée par un nom et une date d'anniversaire. La date d'anniversaire est composée d'un jour, d'un mois et d'une année.

**(a)** Écrire la déclaration d'un type de structure de données `Date` permettant de mémoriser une date d'anniversaire.

```
typedef struct date {
    uint8_t day;
    uint8_t month;
    uint16_t year;
} Date;
```

**(b)** Écrire une fonction permettant de saisir sur l'entrée standard une date au format ISO8601 (YYYY-MM-DD). Cette date est retournée dans le type `Date`. La fonction retourne `0` en cas de succès ou `1` en cas d'erreur de saisie.

```
int read_date(FILE *fp, Date *dt) {
    return fscanf(fp, "%hu-%hhu-%hhu",
        &dt->year, &dt->month, &dt->day) != 3;
}
```

**(c)** Écrire une fonction qui reçoit 2 dates et retourne `0` si elles sont égales `1` si `date1 > date2` ou `-1` si `date1 < date2`.

```
/**
 * Convert a YMD date into an integer allowing comparison.
 */
int date2int(Date dt)
{
    return dt.year * 10000 + dt.month * 100 + dt.day;
}

int compare_date(const Date *a, const Date *b)
{
    if (date2int(*a) > date2int(*b))
        return 1;
    if (date2int(*a) < date2int(*b))
        return -1;
    return 0;
}
```

**(d)** Déclarez un nouveau type basé sur une structure `Birthday` contenant un nom et une date d'anniversaire (`Date`).

```
typedef struct birthday {
    char name[64];
    Date birthday;
} Birthday;
```

**(e)** Déclarez un tableau de `Birthday` de taille `10`. Initialisez la première entrée avec le nom `John Doe` et la date `2000-01-01`.

```
Birthday birthdays[10] = {
    { .name = "John Doe", .birthday = { .year=2000, .month=1, .day=1 } }
};
```

**Exercice 3 : Taille en mémoire**

Quelle est la taille en octets des structures suivantes :

**(a)**
```
struct { int32_t a; int32_t b; };
```

8 bytes

**(b)**
```
struct { char *name; int32_t age; };
```

16 bytes sur une machine 64 bits

**(c)**
```
struct { char k[8]; short s[2]; };
```

12 bytes

**(d)**
```
struct { char a; short b; };
```

4 bytes (à cause de l'alignement)

**(e)**
```
struct { char a; short b; char c[]; };
```

4 bytes (taille de la partie fixe, le membre flexible ne compte pas dans la taille de la structure)

**Exercice 4 : Accès aux membres**

Comment dans `function` afficher sur la sortie standard la valeur de la deuxième entrée `bar` du cinquième élément ?

```
typedef struct { int32_t foo; int32_t bar[8]; } Element;
typedef struct { int32_t count; Element data[100]; } Data;

void function(Data *data) { ... }

int main() { Data data; function(&data); }
```

```
printf("%d\n", data->data[4].bar[1]);
```

**Exercice 5 : Tableau Périodique**

Le tableau périodique des éléments de Dimitri Mendeleev regroupe tous les éléments chimiques connus de l'Homme. Ces éléments sont caractérisés par un nom pouvant comporter jusqu'à 20 lettres, un symbole comportant au maximum 3 lettres, un numéro périodique représentant la position dans le tableau (c'est-à-dire le nombre de protons du noyau), et une valeur logique indiquant si la substance est radioactive ou non. Déclarez les types de données et une variable permettant de stocker les caractéristiques mentionnées ci-dessus pour un nombre variable d'éléments compris entre 0 et 120 éléments inclus.

```
typedef struct element {
    char name[20];
    char symbol[3];
    int period;
    bool is_radioactive;
} Element;

Element[121] elements;
```

**Exercice 6 : Arbre**

Un arbre (en 2D sur une peinture) est composés de branches. Une branche à une longueur, un diamètre, et un angle par rapport à la branche parente. Chaque branche peut avoir jusqu'à 2 branches suivantes (branches enfants). Déclarez les types de données et une variable permettant de stocker les caractéristiques mentionnées ci-dessus pour un arbre.

```
typedef struct branch {
    float length;
    float diameter;
    float angle;
    struct branch *child1;
    struct branch *child2;
} Branch;

Branch tree;
```