

# Structures

## 1 Problématique

## 2 Introduction aux structures

### 2.1 En pratique

#### Exercice 1. Erreur.

Identifiez l'erreur dans chacun des codes suivants. Il n'y a qu'une erreur par code. Vérifiez en compilant les codes.

```
struct Test1
{
    char c;
    int i;
}
```

```
struct Test2
{
    char *c = NULL;
    int i;
};
```

```
struct
{
    char c;
    int i;
}Test3;
```

```
struct Test4
{
    char *c;
    int i
};
```

#### Correction.

- Test1 : il manque le point virgule à la fin.
- Test2 : pas d'initialisation des champs.
- Test3 : le type de la structure doit être placé juste après le mot-clé `struct`.
- Test4 : il manque un point virgule après le champ `i`.

**Exercice 2. Erreur, le retour.**

On suppose que vous avez corrigé correctement les codes précédents. Identifiez l'erreur dans chacune des lignes suivantes. Il n'y a qu'une erreur par ligne. Vérifiez en compilant le code.

```
1 Test1 var;  
2 struct Test2;  
3 struct Test3 v1  
4 struct Test4 v2 = {0};  
5 struct Test1 v3 = {1337, 42};  
6 struct v4 = {NULL, 0};  
7 Test3 v5 = {'c', 0};  
8 struct Test4 v6 = 5;
```

**Correction.**

- Ligne 1 : il manque le mot-clé `struct` au début de la ligne.
- Ligne 2 : il manque un nom de variable.
- Ligne 3 : il manque le point-virgule.
- Ligne 4 : cette initialisation porte uniquement sur le premier champ d'une structure de type `Test4`. Ce dernier est de type `char*`, son initialisation doit donc être `NULL` et non `0`.
- Ligne 5 : `1337` est une valeur trop grande pour un `char`.
- Ligne 6 : il manque le type de la structure après le mot-clé `struct`.
- Ligne 7 : il manque le mot-clé `struct` au tout début.
- Ligne 8 : une initialisation des champs se fait avec des accolades autour des valeurs.

**Exercice 3. Erreur, la suite du retour.**

On suppose que vous avez corrigé correctement les codes précédents. Identifiez l'erreur dans chacune des lignes suivantes. Il n'y a qu'une erreur par ligne. Vérifiez en compilant le code.

```
1 v1.i = c;  
2 v2.c = v2.i;  
3 struct Test1 v7 = {.i, .c = 'z'};  
4 Test2 v8 = {.c = NULL};
```

**Correction.**

- Ligne 1 : la variable `c` n'est pas connue.
- Ligne 2 : `v2` est de type `Test4`, son champ `c` est de type `char*`, son champ `i` est de type `int`. Il y a donc un problème de cast.
- Ligne 3 : aucune valeur n'est affectée au champ `i`.
- Ligne 4 : il manque le mot-clé `struct` au début de la ligne.

## 2.2 Le mot clé typedef

### Exercice 4. Manipulation d'une structure.

- 1) Définissez un type structuré **S1** contenant 2 entiers et un entier court.
- 2) Déclarez une variable **var** de type **S1** dont les valeurs entières sont initialisées à 7 et 3 et l'entier court à 42.
- 3) Affichez l'ensemble des champs de la variable.

#### Correction.

```
// Définition de la structure dans un fichier .h
struct S1_s
{
    int a, b;
    short s;
};
typedef struct S1_s S1;

// Instructions dans un fichier .c
S1 var = { 7,3,42 };
printf("entier a: %d, entier b: %d, entier court s: %hd\n\n", var.a,
    ↪ var.b, var.s);
```

## 2.3 Pointeurs et structures

### Exercice 5. Structure et pointeur.

Reprenez le code rédigé pour l'exercice précédent, et complétez-le avec les instructions suivantes.

- 1) Déclarez une variable **var** de type **S1** ;
- 2) Déclarez un pointeur **p\_var** pointant sur la variable précédemment déclarée.
- 3) Affectez la valeur de votre choix à chacun des champs de votre variable en utilisant uniquement le pointeur.
- 4) Affichez chacun des champs de votre variable en utilisant uniquement le pointeur.

#### Correction.

```
S1 var = { 7,3,42 };
S1 *p_var = &var;
p_var->a = 1;
p_var->b = 3;
p_var->s = 37;
printf("entier a: %d, entier b: %d, entier court s:
    ↪ %hd\n\n", p_var->a, p_var->b, p_var->s);
```

### Exercice 6. Définition d'un étudiant.

- 1) Définissez une structure **Etudiant\_s** comportant les champs demandés dans la section 1 de ce chapitre. Les noms et prénoms seront des tableaux de 128 cases alloués de façon statique. Les noms des champs sont ceux indiqués en gras dans le cours, sans accent.
- 2) Utilisez un **typedef** pour simplifier la déclaration d'une nouvelle variable.
- 3) Définissez une nouvelle variable de type **Etudiant**.

- 4) Demandez à l'utilisateur les informations de la structure et initialisez les champs de votre variable avec ces informations. Le remplissage des champs comportant un tableau n'est pas si compliqué que ça !
- 5) Rédigez une fonction `afficheEtudiant` qui affiche l'ensemble des champs d'une variable de type `Etudiant`.

■ **Correction.** Voir fichiers source.

## 3 Autres types structurés

### 3.1 Les unions

#### Exercice 7. Manipulation d'une union.

Définissez une union nommé `TestUnion` contenant les éléments suivants :

- un caractère `c` ;
- un entier `i` ;
- un tableau `tab` formé de 3 entiers courts.

- 1) Déclarez une variable `var` de type `TestUnion` et attribuez la valeur 0 à chacun des ses champs.
- 2) Affichez l'adresse de la variable `var` ainsi que l'adresse de chacun de ses champs.
- 3) Attribuez la valeur 64 au champ `i` et affichez à nouveau l'ensemble des champs.
- 4) Attribuez la valeur 65 à la case d'indice 2 du champ `tab` et affichez à nouveau l'ensemble des champs.

■ **Correction.** Voir fichiers source.

#### Exercice 8. Un peu d'union.

- 1) Comment définir une union `Calcul` contenant deux entiers courts `s1` et `s2` et un entier classique `n` dont la modification d'un des entiers courts ne doit pas impacter l'autre entier court, mais peut impacter la valeur de `n` ?
- 2) Définissez une union similaire à l'union `Pixel` du cours en utilisant une structure à la place du tableau.

### 3.2 Les énumérations

#### Exercice 9. Quel jour est-on ?.

- 1) Déclarez une énumération `JourSemaine` qui liste les jours de la semaine en partant du dimanche, puis déclarez une variable `aujourd'hui` et affectez lui la valeur du jour actuel.
- 2) Créez une fonction `AfficheJour` prenant une variable du type `JourSemaine` en paramètre et affichant le jour de la semaine correspondant à la valeur de la variable. L'instruction `if` vous est interdite.

■ **Correction.** Voir fichiers source.

#### Exercice 10. Synthèse.

L'objectif de cet exercice est de calculer le nombre de jours restant avant Noël. Les structures vont nous être particulièrement utiles pour stocker la date actuelle de façon appropriée et réaliser un code propre.

- 1) Définissez une structure **Date** contenant trois entiers indiquant une **annee**, un **mois** et un **jour** dans l'année.
- 2) Pour plus de clarté dans le code, modifiez le champ **mois** de votre structure **Date** pour lui attribuer une énumération **Mois** listant les différents mois dans l'année (JAN, FEV, MAR ...).
- 3) Créez la fonction **SaisirDate()** demandant à l'utilisateur la date actuelle et permettant de renvoyer une structure de type **Date** correctement remplie.
- 4) Créez la fonction **JoursAvantNoel()** prenant en paramètres une **Date** et retournant le nombre de jours avant Noël<sup>1</sup>. Pour simplifier, nous pourrions supposer que l'année courante n'est pas bissextile<sup>2</sup>.
- 5) Créez une fonction **Ajouter()** permettant d'ajouter un certain nombre de jours à une date et retournant cette nouvelle date.
- 6) Sachant qu'une année bissextile comporte 366 jours au lieu de 365, et que depuis 1904, les années bissextiles sont les années multiples de 4, créez un programme complet appelant la fonction **time(NULL)** et retournant une structure **Date** à la date actuelle.

**I Correction.** Voir fichiers source.

## 4 En mémoire

### Exercice 11. Structures et état de la mémoire.

Complétez un état de la mémoire pour chacun des codes suivants. N'oubliez pas de barrer les variables qui sont libérées.

```
typedef struct Coord_s
{
    short x,y;
}Coord;
...
Coord points[3];
int i;
for (i=0; i<3; i++)
{
    points[i].x = i*2;
    points[i].y = -i/2;
}
```

Adresse	Nom	Valeur
0x0160		
0x0161		
0x0162		
0x0163		
0x0164		
0x0165		
0x0166		
0x0167		
0x0168		
0x0169		
0x016A		
0x016B		
0x016C		
0x016D		
0x016E		
0x016F		
0x0170		
0x0171		

1. *Indice 1* : Noël ne tombe pas le 21 août cette année.

2. *Indice 2* : vous aurez sûrement besoin de stocker quelque part le nombre de jours présents dans chaque mois pour réaliser correctement l'exercice.

```

typedef union Couleur_u
{
    unsigned char comp[4];
    unsigned int RVBA;
}Couleur;

...
int i;
Couleur pix[2];
pix[0].RVBA = 0x12345678;
pix[1].comp[0] = pix[0].RVBA & 255;
for (i=1; i<4; i++)
    pix[1].comp[i] = pix[1].comp[i-1] + 1;

```

Adresse	Nom	Valeur
0x0160		
0x0161		
0x0162		
0x0163		
0x0164		
0x0165		
0x0166		
0x0167		
0x0168		
0x0169		
0x016A		
0x016B		
0x016C		
0x016D		
0x016E		
0x016F		
0x0170		
0x0171		