Chapitre 10

Les types dérivés

1. Structures

- Une structure permet de regrouper des objets de types hétérogènes.
- Elle permet aussi une meilleure abstraction des données.
- On utilise le mot clé réservé « struct » pour désigner une structure.
- Une déclaration du type « struct » ne définit aucune variable, elle permet de définir un modèle donné (un moule).
- La déclaration doit se terminer par « ; » c'est UNE déclaration.

```
struct Temps {
    int secondes;
    int minutes;
    int heures;
};
```

- La structure « Temps » contient les 3 membres tous des entiers :
    - Les « secondes » sont représentées par le membre « secondes »
    - Les « minutes » sont représentées par le membre « minutes »
    - Les « heures » sont représentées par le membre « heures »

- On peut définir des variables ayant le type de la structure « Temps »

```
struct Temps midi;
```

- On peut l'initialiser et donc initialiser ses membres lors de la définition

```
struct Temps midi{0,0,12}; // OK
```

- Attention : l'ordre des membres dans la structure doit être respecté lors de l'initialisation

```
struct Position {
    char Axe;
    double valeur;
};

struct Position X0{'X',123.45}; // OK

struct Position X1{123.45,'X'}; // Incorrect
```


- On peut déclarer un tableau de structures

```
struct Temps Courses[100];
```

- La variable « Courses » est un tableau dont chaque élément est du type « Temps ».
- La déclaration du type structure peut suivre immédiatement sa définition.

```
struct Temps {
    int secondes;
    int minutes;
    int heures;
} midi, minuit,depart;
```

- À noter que « midi », « minuit » et « depart » sont appelés des instances de la structure « Temps.

- En C++, une structure peut contenir des fonctions membres.
- Ces fonctions ont le rôle de manipuler les membres de données de la structure.

```
#include <iostream>

using namespace std;

struct Temps {
     int secondes;                          }
     int minutes;                           }   Membres de données
     int heures;                            }
     void set(int s, int m, int h);         }   Fonctions membres
     void affiche();                        }
};
```

- La méthode « set » permet d'initialiser les membres de données de la structure « Temps ».
- La méthode « affiche » permet d'afficher « l'heure ».

2. Accès aux membres d'une structure

- On accède aux membres « données » de la structure en utilisant l'opérateur point c.-à-d. « . ».
- Les fonctions membres sont définies soit à l'intérieur de la structure (approche déconseillée) ou bien à l'extérieur. Dans ce cas, la fonction est accessible par l'intermédiaire de l'opérateur unaire de résolution de portée c.-à-d. « **: :** »

```
#include <iostream>

using namespace std;

struct Temps {
     int secondes;
     int minutes;
     int heures;
     void set(int s, int m, int h);
     void affiche();
};

void Temps::set(int s, int m, int h) {
     if (s>=0 && s<=60) secondes=s;
     else cerr << "Erreur paramètre: secondes " << s << "\n";
     if (m>=0 && m<=60) minutes=m;
     else cerr << "Erreur paramètre: minutes " << m << "\n";
     if (h>=0 && h<=24) heures=h;
     else cerr << "Erreur paramètre: heures " << h << "\n";
}
```

```
void Temps::affiche() {
    cout << "L'heure actuelle est : " \
        << heures << " heure(s), " << minutes \
        << " minute(s) et " << secondes << " seconde(s)\n";
}
```

```
int main() {

    Temps t;

    // Initialisation du Temps t

    t.set(0,30,18);

    t.secondes=0;
    t.minutes=30;
    t.heures=18;

    // Affichage du Temps t de deux manières

    cout << "Temps t (h,m,s):(" << t.heures << ", ";
    cout << t.minutes << "," << t.secondes << ")\n";

    t.affiche() ;

    return 0;
}
```

Déclaration de la variable « t » du type « Temps ». L'instance « t » n'a pas été initialisée.

Accès aux membres de données via la fonction membre: set. T=18h, 30m, 00s

Accès direct

3. Pointeurs et structures

- On a montré qu'on pouvait déclarer un pointeur sur une variable donnée. On peut aussi déclarer un pointeur sur une structure.

```
int *ptr;
struct Temps *top;
Temps *minuit;
```

Un pointeur sur une variable du type « int »

Définition d'une variable pointeur sur une structure du type « Temps »

En C++, on peut omettre d'utiliser « struct » puisque on sait d'une précédente définition que « Temps » est une structure

- L'accès à membre particulier de la structure se fait en utilisant le symbole « -> » l'opérateur de pointeur de structure. Attention, il faut initialiser le pointeur avant de l'utiliser, sinon vous allez avoir un comportement indéfini. L'initialiser consiste à lui affecter une structure via une allocation dynamique par exemple.

```
top->set(0,0,0);

top->secondes=0;
top->minutes=0;
top->heures=0;
```

- L'utilisation du symbole « -> » revient à écrire :

```
(*top).set(0,0,0);

(*top).secondes=0;
(*top).minutes=0;
(*top).heures=0;
```

- L'utilisation du symbole « -> » nous évite de recourir à des parenthèses « () » sur chaque référence associée à un membre donné de la structure.

4. Passage de structures

- Une fonction peut prendre comme paramètre une structure.
- Ce passage peut se faire par valeur, par pointeur ou par référence.
- On préférera le passage par pointeur ou par référence.
- En effet, passer un élément par valeur revient à faire une copie locale (dans la fonction) de cet élément.
- Dans le cadre d'une structure, c'est une opération qui peut être coûteuse, voire impossible si la structure occupe une taille mémoire importante.

```
Déclaration pour un passage … … … … …

void test(Temps t);  // par valeur
void test(Temps *t); // par adresse
void test(Temps& t); // par référence
```

```
Appel de la fonction « test » pour un passage … … … …

Temps top;
test(top);  // par valeur
test(&top); // par adresse
test(top);  // par référence
```


5. Constructeur

- Il est possible d'initialiser les paramètres de la structure à l'aide d'une méthode spéciale. Cette méthode porte le nom de « constructeur ».

- Le rôle du constructeur est de construire d'abord un exemplaire de cette structure. On appelle cela une instance de la structure. Par la suite, le constructeur se charge d'initialiser les membres de cette instance avec les valeurs prédéfinies.

- C'est la première méthode appelée au moment de la déclaration « Temps t; ».

- Un constructeur est une méthode qui porte le même nom que la structure et qui ne retourne aucune valeur. On ne met pas « void » comme type de retour.

```
Temps(int s,int m,int h):secondes(s),minutes(m),heures(h){}
```

- Nous allons aborder cette notion plus en détail au moment d'aborder la structure de classe.

6. Structures imbriquées

- Une structure peut contenir n'importe quel type de données.
- Elle peut donc contenir aussi une autre structure => on parle alors de structures imbriquées.

```
struct Position {
     char Axe;
     double valeur;
     void set(char c, double v);
     void affiche();
};
```

```
struct Plan2D {
     Position X;
     Position Y;
} Coordonnees;
```

Structure minimale. Elle peut contenir aussi des fonctions membres.

```
Coordonnees.X.set('X',236.5);
Coordonnees.Y.Axe='Y';

Plan2D *ptr = &Coordonnees;
ptr->Y.valeur=255.9;
```


7. Énumération

- Une énumération est une liste de valeurs entières constantes.
- Par défaut, le compilateur affecte au premier membre la valeur 0 et, chaque membre qui suit le premier prend la valeur du précédent augmenté de 1.
- On peut associer des valeurs à chaque membre si l'on désire.
- Au besoin, on peut imposer des valeurs explicitement à certains (ou à tous) membres.

```
enum Jours {Lun,Mar,Mer,Jeu,Ven,Sam,Dim} ;
// Au lieu de … …
const int Lun=0 ;
const int Mar=0 ;
const int Mer=0 ;
// Etc.

Jours J;

J=5; // 5 => Sam
if (J==Sam || J==Dim) cout << "Bon Week-end!\n"
```

Lun=0;Mar=1;Mer=2
Jeu=3;Ven=4;Sam=5
Dim=6

```
enum Jours {Lun=1,Mar,Mer,Jeu,Ven,Sam,Dim} J;
J=6 ; // 6 => Sam
```

Lun=1;Mar=2;Mer=3
Jeu=4;Ven=5;Sam=6
Dim=7

- D'autres variantes ont été introduites dans les nouveaux standards du langage C++. Ces variantes apportent plus de sécurité et de souplesse.

- Elles sont abordées dans un cours plus avancé.

8. Définition de nouveaux types (typedef)

- La commande « typedef » permet de donner des synonymes à un type pour être utilisés par la suite.

```
typedef char C;
typedef unsigned int MOT;
typedef char* unechaine;
typedef char Champ[50];
```

`C x;`	**x a le type d'un « char »**
`unechaine ptr;`	**ptr est un pointeur vers une chaîne de caractères**
`Champ Tab;`	**Tab est un tableau de 50 caractères**