Information, Calcul et Communication (partie programmation) :

Cours de programmation (C++)
Structures et alias de types

Jamila Sam

Laboratoire d'Intelligence Artificielle
Faculté I&C

EPFL

Vidéos, transparents et quiz

www.coursera.org/learn/initiation-programmation-cpp/

- ☞ Semaine 6 (vidéos 3 et 4)

Jamila Sam
& Jean-Cédric Chappelier

Objectifs du cours d'aujourd'hui

- Rappels sur `struct`
  - ☞ nouveau **type**
- Rappels sur `typedef`
- Etude de cas

Jamila Sam
& Jean-Cédric Chappelier

Données structurées

Âge
20
35
26
38
22

Nom	Taille	Âge	Sexe
Dupond	1.75	41	M
Dupont	1.75	42	M
Durand	1.85	26	F
Dugenou	1.70	38	M
Pahut	1.63	22	F

Les tableaux permettent de représenter des structures de données **homogènes**, c'est-à-dire des listes constituées d'éléments qui sont tous du **même type**.

Les **structures** permettent de regrouper des types **hétérogènes** (mais en nombre limité, connu au préalable !).

Déclaration d'un nouveau type

Exemples :

```
struct Personne {
  string nom;
  double taille;
  int age;
  char sexe;
};
```

déclare un nouveau type, `Personne`, comme une `struct` composée de quatre **champs** : un de type `string`, un autre de type `double`, un troisième de type `int` et un dernier de type `char`.

Autre exemple :

```
struct Complexe {
  double x;
  double y;
};
```

Initialisation d'une structure

Les structures peuvent être initialisées avec la syntaxe suivante :

```
struct Nom {
 type1 champ1;
 type2 champ2;
 // ....
};

Nom identificateur = { val1, val2, ... };
```

On peut également utiliser cette syntaxe pour l'affectation.

Accès aux champs d'une structure

On peut accéder aux champs d'une structure en utilisant la syntaxe suivante :

```
structure.champ
```

Exemples :

```
untel.taille = 1.75;

++(untel.age); // un an de plus !

cout << untel.nom << endl;
```


Jamila Sam
& Jean-Cédric Chappelier

Affectation de structures

Une variable de type composé `struct` peut être **directement affectée** par une variable du même type

Exemple :

```
Personne p1 = { "Durand", 1.75, 20, 'M' };
Personne p2;
p2 = p1;
```

La valeur de chaque champ de `p1` est affectée au champ correspondant de `p2`

☞ L'instruction `p2=p1` est équivalente à la séquence d'instructions

```
p2.nom    = p1.nom;        p2.taille = p1.taille;
p2.age    = p1.age;        p2.sexe   = p1.sexe;
```

Note : l'affection (=) est la seule opération que l'on peut faire globalement sur les `struct`.

On **NE** peut **NI** les comparer (`p1 == p2`),
NI les afficher (`cout << p1`) globalement.

Fonction à plusieurs valeurs de retour

On sait que les fonctions ne peuvent retourner qu'une seule valeur.

Comment faire lorsque l'on veut « retourner » *plusieurs* valeurs avec une fonction ?

Solutions :

1. renvoyer une **structure** contenant les valeurs à retourner ;
2. **passer** les « variables retour » **par référence** et les affecter à l'intérieur de la fonction ;
3. renvoyer un **tableau dynamique** (`vector`), si les valeurs à retourner sont de même type (homogène) ;
4. combiner 1 et 3 : structure avec champs `vector` ou (au choix) `vector` de structures (comme dans l'exemple précédent)

Jamila Sam
& Jean-Cédric Chappelier

Les structures

Déclaration du type correspondant :

```
struct Nom_du_type {
        type1 champ1 ;
        type2 champ2 ;
        ...
};
```

Déclaration d'une variable :

```
Nom_du_type identificateur;
```

Déclaration/Initialisation d'une variable :

```
Nom_du_type identificateur = { val1, val2, ... };
```

Accès à un champ donné de la structure :

```
identificateur.champ
```

Affectation globale de structures :

```
identificateur1 = identificateur2
```

Alias de types

Pour des types composés complexes, dont l'utilisation directe est difficile, on peut utiliser la commande `typedef` pour

donner un autre nom (alias) à ce type

Syntaxe :

```
typedef type alias;
```

où `alias` est le nouveau nom de type et `type` un type élémentaire ou composé.

Exemple :

```
typedef vector<vector<double> > Matrice;
```

Un tableau bidimensionnel d'entiers pourra alors être déclaré plus simplement (et plus lisiblement) par :

```
Matrice rotation(3, vector<double>(3, 1.0)) ;
```

Mieux :

```
typedef vector<double> Vecteur;
typedef vector<Vecteur> Matrice;
Matrice rotation(3, Vecteur(3, 1.0));
```

Etude de cas

- Trêve de calcul d'IMC … et pourquoi pas une bonne fondue ?

Pour préparer le prochain cours

Vidéos et quizz du MOOC semaine 7 (pointeurs et références)

Jamila Sam
& Jean-Cédric Chappelier