Support de cours

Cours:

Initiation à la programmation (en C++)

Vidéo:

Structures (partie 2)

Concepts (extraits des sous-titres générés automatiquement) :

Nouveau type. Nom de la variable. Valeurs des différents champs. Nom du type. Dehors des fonctions. Champ x. Syntaxe suivante. Accolade ouvrante. Façon suivante. Deuxième exemple. Moment de cette déclaration. Intérieur d'une fonction. Définition de la structure. Variables. Champ.

vers la recherche de séquences vidéo
(dans Initiation à la programmation (en C++).)

vers la vidéo

EPFL

Structures
(Partie 2)

Initiation à la programmation (C++)

Vincent Lepetit, Jean-Cédric Chappelier et Jamila Sam

...

notes

résumé

0m 0s

Initialisation

EPFL

Les variables de type structure peuvent être initialisées avec la syntaxe suivante :

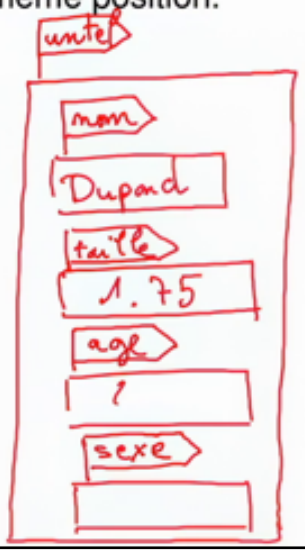
Type identificateur = { val_1, val_2, ...};

où chaque *val_i* est de type *type_i*, correspondant au champ de même position.

Exemple :

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

Personne untel = { "Dupond", 1.75, 41, 'M' };
```



C++11 On peut également utiliser cette syntaxe pour l'affectation :

```
untel = { "Dupond", 1.75, 41, 'M' };
```

Une structure est donc un nouveau type, que je peux utiliser pour déclarer des variables, et une déclaration va se faire de la même façon qu'avec n'importe quel autre type, c'est-à-dire que je vais commencer par le nom du type, suivi du nom de la variable. Par exemple, j'ai repris ici le code qui définit la structure "Personne", et ici, je déclare une variable qui s'appelle "untel" et qui sera de type "Personne". Attention, la définition de la structure doit s'écrire en dehors des fonctions, avant de l'utiliser pour déclarer des variables. Par-contre, comme avant, la déclaration des variables de ce nouveau type doit se faire à l'intérieur d'une fonction. Par exemple, on écrira en dehors de toute fonction, puis on peut définir la variable "untel" à l'intérieur d'une fonction, par exemple la fonction "main", comme ceci. Cette variable "untel", je peux la représenter de la façon suivante : elle aura un champ qui s'appelle "nom", un champ qui s'appelle "taille", etc. Dans ce deuxième exemple, je déclare tout d'abord une structure qui s'appelle "complexe" avec un champ x et un champ y, et je déclare ici une nouvelle variable, qui s'appelle z, qui sera de type "complexe", et donc z va contenir deux champs : un champ x et un champ y. Comme avec les autres types, on peut initialiser une variable de type "structure". Au moment de cette déclaration, ça va se faire avec la syntaxe suivante. Donc après le type de la structure, le nom de la variable : on va mettre le signe "=" avec, entre accolade ouvrante et accolade fermante, les valeurs des différents champs qu'on veut donner à la variable. Et après l'accolade fermante, on va avoir ici un point-virgule. Par exemple, j'ai repris ici une nouvelle fois ma structure "Personne" et déclaré une variable "untel" de type "Personne". Je peux

notes

résumé

0m 1s

Initialisation

EPFL

Les variables de type structure peuvent être initialisées avec la syntaxe suivante :

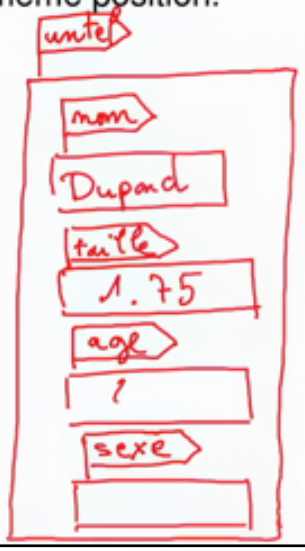
Type identificateur = { val_1, val_2, ...};

où chaque *val_i* est de type *type_i*, correspondant au champ de même position.

Exemple :

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

Personne untel = { "Dupond", 1.75, 41, 'M' };
```



C++11 On peut également utiliser cette syntaxe pour l'affectation :

```
untel = { "Dupond", 1.75, 41, 'M' };
```

représenter "untel" de cette façon. Cette déclaration initialise également les champs de la variable "untel" ; Dupond va aller dans le premier champ, qui est le champ "nom", c'est-à-dire que le champ "nom" va contenir la valeur "Dupond", le champ "taille" va contenir la valeur "1.75",

notes

résumé

Accès aux champs d'une structure

EPFL

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

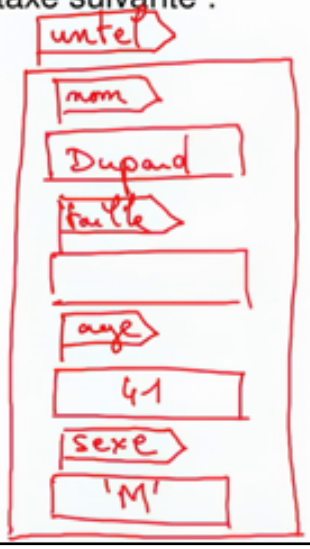
→ structure.champ

Exemples :

```
untel.taille = 1.75;

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

cout << untel.sexe << endl;
```



le champ "âge", la valeur "41" et le champ "sexe", le caractère "M". Par ailleurs, en C++2011, on peut également utiliser cette syntaxe en dehors d'une initialisation pour une affectation et on va obtenir les mêmes valeurs aux mêmes endroits. Notez bien qu'en C++ avant 2011, on ne pouvait pas utiliser cette syntaxe pour l'affectation et il n'y avait pas de moyen simple d'affecter tous les champs d'une structure à des valeurs littérales. On sait donc initialiser les champs d'une variable de type "structure", c'est-à-dire donenr une valeur à ces champs, au moment de la déclaration de la variable, encore faut-il pouvoir accéder à ces valeurs, ça va se faire en respectant la syntaxe suivante, c'est-à-dire qu'on va commencer par écrire le nom de la variable de type "structure", suivi d'un point, suivi du nom du champ. Par exemple, si je reprends ma variable "untel" du transparent précédent, qui était de type "Personne" et que je peux représenter de la façon suivante, cette instruction est une affectation

notes

résumé

3m 13s

qui va mettre la valeur "1,75" dans le champ "taille" de la variable "untel", c'est-à-dire mettre la valeur "1,75" à cet endroit dans ma représentation de la variable "untel". Cette instruction va ajouter 1 au champ "âge" de "untel", c'est-à-dire changer ce "41" en "42". Notez au passage que je suis obligé d'utiliser des parenthèses ici : je ne peux pas écrire simplement "++untel.age" car l'opérateur "++" est prioritaire sur l'opérateur "." Et cette dernière instruction va afficher la valeur contenue dans le champ "sexe" de la variable "untel", c'est-à-dire le caractère "M". le caractère "M".

notes

résumé

4m 25s