Support de cours

Cours:

Initiation à la programmation (en C++)

Vidéo:

Structures (partie 1)

Concepts (extraits des sous-titres générés automatiquement) :

Champs de la structure. Structures de données homogènes. Âges de plusieurs personnes. Données structurées. Types différents. Nouveau type. Déclaration de ces champs. Nom de la personne. Forme d'un caractère m. Telle déclaration. Différentes utilisations. Type. Chaînes de caractère. Tel cas. Tableau de structures.

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

vers la vidéo

EPFL

Structures
(Partie 1)

Initiation à la programmation (C++)

Vincent Lepetit, Jean-Cédric Chappelier et Jamila Sam

...

notes

résumé

0m 0s

Lors des leçons précédentes, nous avons vu les tableaux et les chaînes de caractère.

notes

résumé

0m 1s

Données structurées

Un programme peut avoir à représenter des **données structurées**, par exemple :

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

Dans cette vidéo nous allons voir les structures, qui sont une autre façon de représenter des données.

notes

résumé

0m 5s

Données structurées

EPFL

Un programme peut avoir à représenter des **données structurées**, par exemple :

Âge	Nom	Taille	Âge	Sexe
20	Dupond	1.75	41	M
35	Dupont	1.75	42	M
26	Durand	1.85	26	F
38	Dugenou	1.70	38	M
22	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*.
Exemples :

```
vector<int> ages;
```

```
array<int, 5> ages;
```

QUID des données *hétérogènes* (c'est-à-dire non homogènes) ?

☞ On les regroupe dans un type composé : les **structures**

Un programme peut avoir besoin de stocker ce qu'on appelle des données structurées, par exemple un programme peut avoir besoin de stocker les âges de plusieurs personnes, comme ce qui est représenté ici, et nous avons vu lors d'une séance précédente que dans un tel cas, il faut utiliser un tableau, c'est-à-dire un tableau qui serait déclaré de cette façon-ci, en utilisant le type "vector", si on ne sait pas combien de personnes le programme doit considérer ; ou déclarer de cette façon-ci, en utilisant le type "array" si par exemple on sait que l'on a cinq personnes. Les tableaux sont des structures de données homogènes, c'est-à-dire que toutes les valeurs contenues dans un même tableau sont toutes de même type, le type "int" dans le cas de cet exemple. Mais on peut avoir aussi besoin de considérer des données hétérogènes, c'est-à-dire des données ayant des types différents, par exemple si on veut considérer les données relatives à une personne, comme ici, où les données seraient par exemple d'abord le nom de la personne, qui serait une chaîne de caractères, la taille de la personne, qui serait de type "double", son âge, qui serait de type "int" et son sexe représenté sous la forme d'un caractère M ou F. Ici, on ne peut pas utiliser de tableau pour stocker toutes ces données puisqu'elles sont de type différent, par contre on peut les regrouper dans

notes

résumé

0m 14s

A quoi servent les structures ?

EPFL

On utilise les **structures** lorsque l'on souhaite regrouper des types

Elles servent à :

- représenter des entités qui doivent être décrites avec plusieurs données
  Par exemple :
  - une personne peut être définie avec son prénom, son nom, sa date de naissance, ...
  - une date est définie par un jour, un mois, et une année

  Elles facilitent la manipulation de telles entités en regroupant les données.
- faire retourner plusieurs valeurs à une fonction
- simplifier la conception et l'écriture des programmes (regroupements conceptuels)

un nouveau type, qu'on appelle un type "structure". Nos structures nous permettraient de représenter les données d'une ligne. On peut ensuite faire un tableau de structures pour représenter l'ensemble de ces données relatives à un ensemble de personnes. De façon plus générale, on utilise des structures quand on souhaite regrouper des données ayant des types ou des natures différents ; elles vont avoir différentes utilisations. Tout d'abord, on va pouvoir représenter des entités qui doivent être définies avec plusieurs données, par exemple une telle entité pourrait être une personne qui, suivant les besoins du programme, serait définie par son prénom, son nom, sa date de naissance.

notes

résumé

1m 37s

Quelques exemples de structures

```
struct Date
{
  int jour;
  int mois;
  int annee;
};
```

```
struct Etudiant
{
  string nom;
  string section;
  vector<Cours> inscriptions;
  double moyenne;
};
```

```
struct Particule
{
  array<double, 3> position;
  array<double, 3> vitesse;
  double masse;
  double charge;
};
```

Une autre entité pourrait être une date, qui serait définie par un numéro de jour, de mois et d'année. Regrouper ces données facilite la manipulation de telles entités. Une deuxième utilisation possible des structures est de faire retourner plusieurs valeurs à une fonction, que nous allons détailler comment par la suite. Et enfin, une troisième utilisation possible, qui généralise la première utilisation, est de simplifier la conception et l'écriture des programmes. Avant d'entrer dans les détails, voici quelques exemples de structures.

notes

résumé

2m 25s

Quelques exemples de structures

```
struct Date
{
  int jour;
  int mois;
  int annee;
};
```

Date aujourdhui;

```
struct Etudiant
{
  string nom;
  string section;
  vector<Cours> inscriptions;
  double moyenne;
};
```

```
struct Particule
{
  array<double, 3> position;
  array<double, 3> vitesse;
  double masse;
  double charge;
};
```

Cette structure "Date" réunit un jour, un mois et une année. Une telle déclaration définit donc un nouveau type utilisable dans un programme. Ce type va nous permettre de déclarer des variables, par exemple comme ceci qui est la déclaration d'une variable qui s'appelle "aujourd'hui" de type "date"

notes

résumé

3m 0s

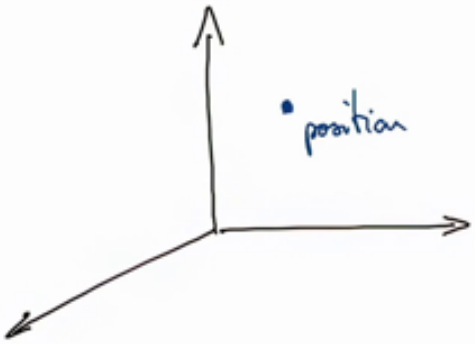
Quelques exemples de structures

```
struct Date
{
  int jour;
  int mois;
  int annee;
};
```

Date aujourdhui;

```
struct Etudiant
{
  string nom;
  string section;
  vector<Cours> inscriptions;
  double moyenne;
};
```

```
struct Particule
{
  array<double, 3> position;
  array<double, 3> vitesse;
  double masse;
  double charge;
};
```



et qui va pouvoir avoir des valeurs concrètes pour le jour, le mois et l'année. Cette structure "Etudiant" regroupe le nom d'un étudiant avec le nom de sa section sous la forme de chaîne de caractères, ainsi que ses inscriptions au cours, sous la forme d'un tableau, où les éléments seraient de type "cours", où "cours" serait une structure qui devrait être déclarée plus haut. La dernière donnée contenue dans la structure serait la moyenne des notes de l'étudiant. Cette structure "particule" serait utile à un programme de simulation de physique, puisqu'elle regroupe les valeurs relatives à une particule élémentaire, avec tout d'abord la position de la particule sous la forme d'un vecteur à trois dimensions,

notes

résumé

3m 25s

Déclaration d'une structure

EPFL

Pour déclarer un nouveau **type** « structure », on utilise la syntaxe suivante :

```
struct Nom_du_type {
    type_1 identificateur_1 ;
    type_2 identificateur_2 ;
    ...
} ;
```

Champs

où

Nom_du_type est le nom que vous souhaitez donner à votre type structuré,
et les

type_i identificateur_i sont les *déclarations des types* et *identificateurs* des **champs** de la structure.

La vitesse de la particule qui aurait également trois coordonnées, ainsi que la masse et la charge de la particule, qui sont toutes les deux de type "double". Une structure est donc un nouveau type, qu'un programmeur va définir et déclarer en respectant la syntaxe suivante, c'est-à-dire en commençant par le mot-clé "struct", suivi du nom qu'il veut donner à la structure, suivi entre accolades ouvrante et fermante, de la liste de ce qu'on appelle les champs de la structure et qui sont les données que regroupe la structure. Et vous pouvez remarquer que la déclaration de ces champs

notes

résumé

4m 13s

Déclaration d'une structure

Exemples :

```
struct Personne {
  string nom;
  double taille;
  int    age;
  char   sexe; // 'M' ou 'F'
};
```

déclare un nouveau **type**, `Personne`, comme une `structure` 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;
};
```

ressemble beaucoup à la déclaration d'une variable, puisqu'une déclaration commence par le type qu'on veut donner au champ, suivi de son identificateur ou son nom, suivi d'un point-virgule. Notez enfin qu'il faut un point-virgule après l'accolade fermante. Une structure : le seul cas où un point-virgule apparaît après une accolade fermante. Par exemple ce que défnit une nouvelle structure donc un nouveau type qui s'appelle "Personne" et qui serait utile pour définir le type des éléments du tableau qu'on a vu au début du cours, puisqu'elle regroupe le nom, la taille, l'âge et le sexe d'une personne ; Cette autre structure s'appelle "complexe" et regroupe deux champs :

notes

résumé

5m 1s

un champ x et un champ y et serait utile pour représenter des valeurs complexes, par opposition à des valeurs réelles, qui se noteraient en mathématiques : x+iy.

notes

résumé

5m 49s

Déclaration d'une structure (2)

Note : Les types des champs d'une structure peuvent aussi être des *types composés*, par exemple des tableaux ou des structures.

Exemple :

```
struct Simple {
  int    souschamp1;
  double souschamp2;
};

struct Compliquee {
  vector<double> champ1;
  int            champ2;
  Simple         champ3;
};
```

Comme les éléments de tableau, les champs d'une structure ne sont pas limités à être de type scalaire comme "int" ou "double". Les champs d'une structure peuvent être des tableaux, voire des structures.

notes

résumé

6m 1s

Par exemple, je déclare ici un type qui s'appelle "Compliquee", où le deuxième champ est de type "int", mais le premier champ est un tableau de "double". Et le troisième champ est de type "simple", qui a été déclaré ici, qui contient lui-même deux champs. Le premier de type "int", le deuxième de type "double". le deuxième de type "double".

notes

résumé

6m 13s