Support de cours

Cours:

## Initiation à la programmation (en C++)

Vidéo:

## Structures (partie 4)

Concepts (extraits des sous-titres générés automatiquement) :

**Tel cas. Solutions possibles. Deuxième possibilité. Troisième possibilité. Type de retour de la fonction. Champs vecteurs. Type d'une variable. Première solution. Données homogènes. Deuxième solution. Intérieur de la fonction. Troisième solution. Différentes valeurs. Cas de cette fonction. Vecteurs de structure.**

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

vers la vidéo

EPFL

# Structures
## (Partie 4)

Initiation à la programmation (C++)

Vincent Lepetit, Jean-Cédric Chappelier et Jamila Sam

...

notes

résumé

0m 0s

## Fonction à plusieurs valeurs de retour

EPFL

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

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

Par exemple, faire retourner le quotient et le reste à la fonction `division_euclidienne(a, b)` ?

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)

J'ai dit que les structures étaient également utiles pour permettre aux fonctions de renvoyer plusieurs valeurs, par exemple je peux avoir envie d'écrire une fonction "division_euclidienne", qui renverrait à la fois le quotient et le reste de deux valeurs passées en argument ; et bien dans un tel cas, j'ai plusieurs solutions possibles : je peux donc faire renvoyer, par la fonction, une structure qui va contenir les différentes valeurs à retourner. Une deuxième possibilité est de passer des variables par référence et de les affecter à l'intérieur de la fonction aux valeurs que la fonction doit calculer. Nous allons voir un exemple sur le transparent suivant. Une troisième possibilité, dans le cas où toutes les valeurs sont de même type, est de faire renvoyer, par la fonction, un tableau dynamique qui va contenir ces valeurs. Dans des cas plus complexes, on peut avoir aussi envie de combiner les solutions 1 et 3, c'est-à-dire renvoyer une structure, avec des champs vecteurs, qui vont contenir plusieurs données homogènes ou des vecteurs de structure contenant

notes

résumé

0m 1s

## Fonction à plusieurs valeurs de retour

EPFL

Exemple :

1.
```
struct Resultat {
  int quotient;
  int reste;
};
Resultat division_euclidienne(int dividende, int diviseur);
```

2.
```
void division_euclidienne(int dividende, int diviseur,
                          int& quotien, int& reste);
```

3.
```
array<int, 2> division_euclidienne(int dividende, int diviseur);
```

ou

```
vector<int> division_euclidienne(int dividende, int diviseur);
```

des valeurs hétérogènes. Par exemple, dans le cas

notes

résumé

1m 13s

## Fonction à plusieurs valeurs de retour

EPFL

Exemple :

1.
```
struct Resultat {
  int quotient;
  int reste;
};
Resultat division_euclidienne(int dividende, int diviseur);
```

2.
```
void division_euclidienne(int dividende, int diviseur,
                          int& quotien, int& reste);
```

3.
```
array<int, 2> division_euclidienne(int dividende, int diviseur);
```

ou

```
vector<int> division_euclidienne(int dividende, int diviseur);
```

de cette fonction "division_euclidienne", qui aurait deux paramètres : un pour le dividende, l'autre pour le diviseur, la première solution consisterait tout d'abord à déclarer une structure, qu'on a appelée ici "Résultat", qui contiendrait un champ pour le quotient et un champ pour le reste, et définir le type de retour

notes

résumé

1m 15s

# Fonction à plusieurs valeurs de retour

EPFL

Exemple :

1.
```
struct Resultat {
  int quotient;
  int reste;
};
Resultat division_euclidienne(int dividende, int diviseur);
```

2.
```
void division_euclidienne(int dividende, int diviseur,
                          int& quotien, int& reste);
```

3.
```
array<int, 2> division_euclidienne(int dividende, int diviseur);
```

ou

```
vector<int> division_euclidienne(int dividende, int diviseur);
```

de la fonction "division_euclidienne" comme "Resultat". Dans un tel cas, j'appellerais ma fonction "division_euclidienne"

notes

résumé

1m 37s

## Fonction à plusieurs valeurs de retour

EPFL

Exemple :

1.
```
struct Resultat {
  int quotient;
  int reste;
};
Resultat division_euclidienne(int dividende, int diviseur);
```

Resultat res (division_euclidienne(13,6));

2.
```
void division_euclidienne(int dividende, int diviseur,
                          int& quotien, int& reste);
```

int

3.
```
array<int, 2> division_euclidienne(int dividende, int diviseur);
```

ou

```
vector<int> division_euclidienne(int dividende, int diviseur);
```

de la façon suivante : "Résultat", pour définir le type d'une variable qui recevrait le résultat, en lui passant deux valeurs, pour le dividende et le diviseur. La deuxième solution consiste à ajouter deux paramètres passés par référence et utiliser ma fonction va se faire de la façon suivante. Je vais commencer par déclarer

notes

résumé

1m 45s

# Fonction à plusieurs valeurs de retour

EPFL

Exemple :

1.
```
struct Resultat {
  int quotient;
  int reste;
};
Resultat division_euclidienne(int dividende, int diviseur);
```

Resultat res (division_euclidienne(13,6));

2.
```
void division_euclidienne(int dividende, int diviseur,
                          int& quotien, int& reste);
```

int q, r;
division_euclidienne(13,6,q,r);

3.
```
array<int, 2> division_euclidienne(int dividende, int diviseur);
```

ou

```
vector<int> division_euclidienne(int dividende, int diviseur);
```

deux variables supplémentaires, qui vont me servir à stocker les valeurs calculées par la fonction. Et je peux ensuite appeler ma fonction de cette façon. La troisième solution consiste à définir le type de retour de la fonction

notes

résumé

2m 8s

## Fonction à plusieurs valeurs de retour

EPFL

Exemple :

1.
```
struct Resultat {
  int quotient;
  int reste;
};
Resultat division_euclidienne(int dividende, int diviseur);
```

2.
```
void division_euclidienne(int dividende, int diviseur,
                          int& quotien, int& reste);
```

3.
```
array<int, 2> division_euclidienne(int dividende, int diviseur);
```

ou

```
vector<int> division_euclidienne(int dividende, int diviseur);
```

comme un tableau entier à deux éléments. Et ma fonction s'appellerait alors ainsi.

notes

résumé

2m 25s

On a utilisé ici le type "array" mais on pourrait utiliser aussi le type "vector", puisque le type "vector" est la forme la plus générale de tableau. En résumé, les structures permettent de regrouper plusieurs données sous un même type et servent à représenter des entités comme des personnes ou des dates, à faire retourner aux fonctions plusieurs valeurs, et plus généralement à simplifier la conception et l'écriture de programmes. Donc je vous invite maintenant à répondre au quizz pour tester les connaissances que vous avez acquises sur cette nouvelle leçon. sur cette nouvelle leçon.

notes

résumé

2m 36s