Support de cours

Cours:

Initiation à la programmation (en C++)

Vidéo:

Introduction

Concepts (extraits des sous-titres générés automatiquement) :

Score des joueurs. Cas des statistiques. Tel tableau. Taille du tableau. Différents joueurs. Type double. Écart de ces scores. Score moyen. Type bool. Certaine fonction. Nombres entiers. Âges des différents étudiants. Deuxième joueur. Tableaux. Programme de jeu.

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

vers la vidéo

EPFL

Tableaux : introduction

Initiation à la programmation (C++)

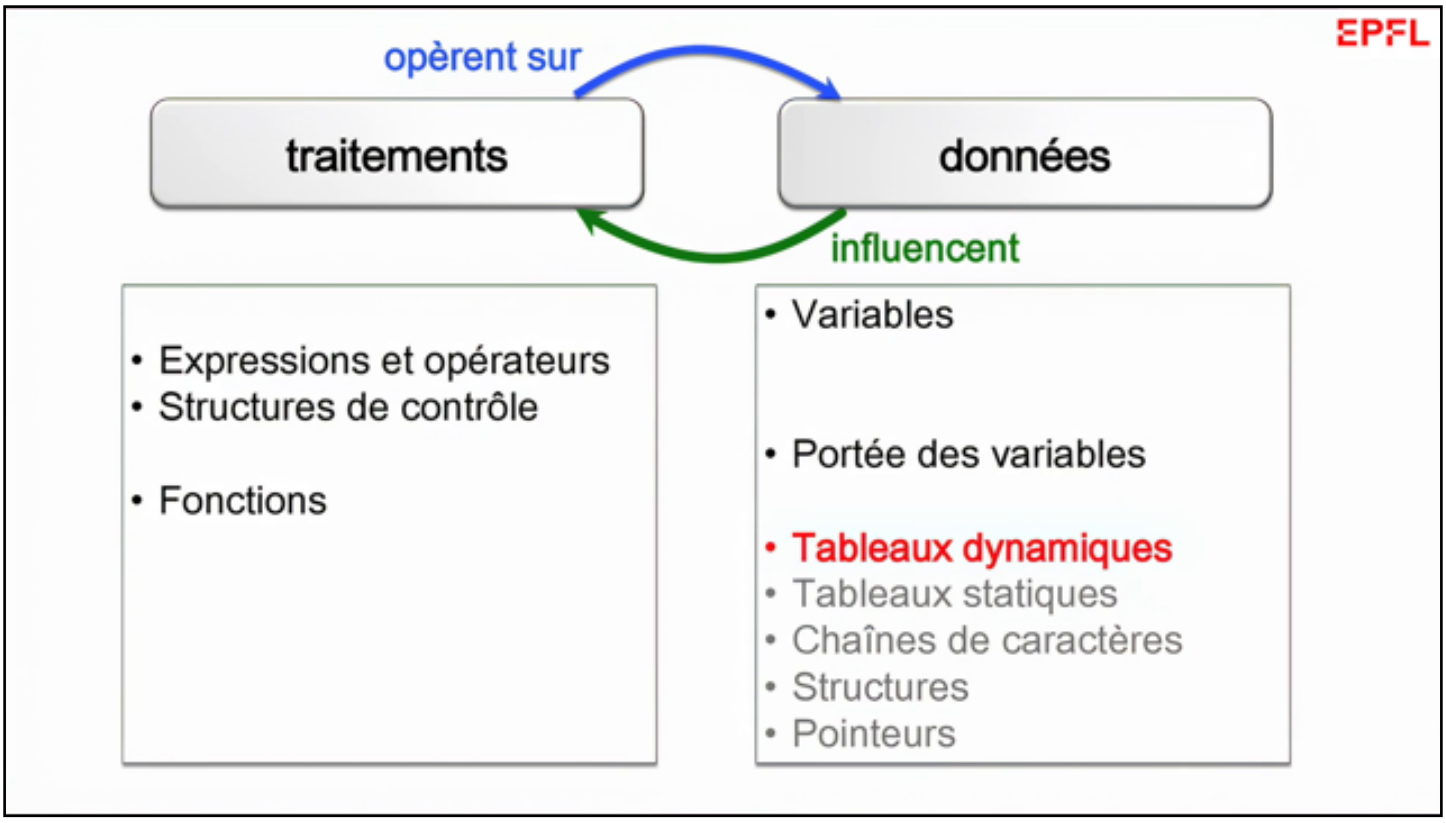
Vincent Lepetit, Jean-Cédric Chappelier et Jamila Sam

...

notes

résumé

0m 0s



Dans cette vidéo-ci, ainsi que dans les suivantes,

notes

résumé

0m 1s

Exemples de données structurées

EPFL

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

- tableaux
- structures de données hétérogènes
  (par exemple, « un enregistrement » dans le tableau de droite ci-dessus)
- chaînes de caractères
  (par exemple, le « nom »)
- ...

nous allons moins nous focaliser sur les traitements, mais plus nous focaliser sur les données. Comme vous le savez dans un langage de programmation, les données sont représentées par des variables. Et en C++, les variables sont typées. A ce stade du cours, nous avons surtout présenté les types int, pour représenter des nombres entiers, le type double, pour représenter des grandeurs réelles, et nous avons aussi évoqué le type bool pour représenter des valeurs de vérité. Ces types, int, double, bool, que l'on appelle les types élémentaires, servent à représenter des données simples du monde, comme des dimensions, des sommes, des expressions logiques. Mais comment faire pour des données plus complexes, plus structurées ? Comment représenter dans le programme qui gère le cours que vous êtes en train de regarder, les âges des différents étudiants? Un autre exemple, si on avait un programme qui doit gérer des statistiques sur la population, comment seraient représentées ces statistiques ? Comment seraient représentés ce que l'on appelle donc des tableaux ? Ou dans le cas des statistiques,

notes

résumé

0m 5s

Petit exemple introductif

Supposons que l'on souhaite écrire un programme de jeu à plusieurs joueurs.

Score	Écart à la moyenne
1000	-1860
1500	-1360
2490	-370
6450	3590
...	...

comment serait représentée une personne dans un tel tableau ? La réponse est la notion de structure de données. Comment d'ailleurs représenteriez-vous simplement le nom d'une de ces personnes avec les types que l'on a vus jusqu'à maintenant ? Vous ne sauriez pas le faire. Cette séquence vidéo va s'intéresser aux tableaux. Nous verrons plus tard dans d'autres séquences vidéo, comment représenter les chaînes de caractères, et puis comment représenter des structures, comment représenter une ligne dans ce tableau. Mais commençons donc par les tableaux.

notes

résumé

1m 13s

Solution avec les moyens actuels

```
int score1(calcule_score(....));
int score2(calcule_score(....));

// Calcul de la moyenne
int moyenne_joueurs(moyenne(score1, score2));

// Affichages
cout << "Score          Ecart Moyenne" << endl;
cout << score1 << "     " << score1 - moyenne_joueurs << endl;
cout << score2 << "     " << score2 - moyenne_joueurs << endl;
```

Comment passer à plus de joueurs ?

Supposons que l'on veuille créer un programme de jeu et que l'on ait à gérer le score des joueurs, on voudrait par exemple, que pour les différents joueurs, on puisse afficher leur score, et puis par exemple, afficher l'écart de ces scores au score moyen. Comment procéderions-nous ? Commençons modestement par nous intéresser à un programme qui aurait deux joueurs. Donc ce que l'on ferait, bien sûr, c'est que l'on introduirait une variable représentant le score pour chacun des deux joueurs. Disons typiquement que les scores sont ici des entiers, que l'on initialiserait avec une certaine fonction ici. Cette fonction calculerait le score et le retournerait. On ferait le calcul du score comme ceci pour le premier joueur. Le calcul du score pour le deuxième joueur se ferait de façon analogue avec ses paramètres spécifiques. Et ensuite, comme on veut l'écart à la moyenne, on va calculer la moyenne des deux joueurs en utilisant la fonction moyenne que l'on avait déjà illustrée dans des vidéos précédentes, qui prendrait en paramètre les scores de nos deux joueurs. On stockerait le résultat dans une variable moyenne_joueurs de type entier. Et enfin, on pourrait afficher le tableau des scores, en affichant le premier score et l'écart à la moyenne, et en affichant ensuite le deuxième score et son écart à la moyenne. Deux joueurs, c'est déjà pas mal, mais la question c'est,

notes

résumé

1m 41s

Solution avec les moyens actuels (2)

```
int score1(calcule_score(...));
int score2(calcule_score(....));
int score3(calcule_score(....));
int score4(calcule_score(....));
int score5(calcule_score(....));

// Calcul de la moyenne
int moyenne_joueurs(moyenne(score1, score2, score3, score4, score5));

// Affichages
cout << "Score          Ecart Moyenne" << endl;
cout << score1 << "     " << score1 - moyenne_joueurs << endl;
cout << score2 << "     " << score2 - moyenne_joueurs << endl;
cout << score3 << "     " << score3 - moyenne_joueurs << endl;
cout << score4 << "     " << score4 - moyenne_joueurs << endl;
cout << score5 << "     " << score5 - moyenne_joueurs << endl;
```

comment améliorer notre programme pour passer à plusieurs joueurs ? Il suffirait pour cela simplement d'utiliser plusieurs variables. Donc par exemple, si l'on voulait passer à cinq joueurs, on utiliserait alors typiquement cinq scores, un pour chacun des joueurs, on calculerait le score de chacun des joueurs avec leurs différents paramètres. Puis on calculerait la moyenne de tous ces joueurs en ayant ici cette fois-ci changé la fonction moyenne et en passant les cinq scores à notre nouvelle fonction moyenne, puis on écrirait comme ça le tableau des scores.

notes

résumé

3m 5s

Solution avec les moyens actuels (3)

```
int score1(calcule_score(....));
int score2(calcule_score(....));
int score3(calcule_score(....));
int score4(calcule_score(....));
int score5(calcule_score(....));

// Calcul de la moyenne
int moyenne_joueurs(moyenne(score1, score2, score3, score4, score5));

// Affichages
cout << "Score          Ecart Moyenne" << endl;
for(int i(1); i <= 5; ++i) {
  cout << scorei << "     " << scorei -  moyenne_joueurs << endl;
}
```

Ici, tout bon programmeur pense tout de suite

notes

résumé

3m 37s

Solution avec les moyens actuels : limites

Mais

1. comment l'écrire (`scorei` n'est pas correct) ?
2. comment faire si on veut considérer 100, 1000... joueurs ?
3. comment faire si le nombre de joueurs n'est pas connu au départ ?

naturellement à une boucle, et va donc modifier cette partie là du programme pour faire une boucle ici, qui va aller du premier joueur au cinquième joueur, de un en un pour tous les joueurs, et qui va afficher le score de chacun des joueurs, et l'écart à la moyenne pour chacun des joueurs. La question c'est comment le faire vraiment concrètement.

notes

résumé

3m 40s

Car déjà premièrement, l'écriture comme ceci, scorei, du programme précédent, n'est pas valable. scorei, c'est un nom de variable et i ne peut pas prendre différentes valeurs, ce n'est pas score1, score2. Donc c'est quelque chose qui ne fonctionne pas.

notes

résumé

4m 5s

Solution avec les moyens actuels : limites

Mais

1. comment l'écrire (`score1` n'est pas correct) ?
2. comment faire si on veut considérer 100, 1000... joueurs ?
3. comment faire si le nombre de joueurs n'est pas connu au départ ?

Ensuite, comment faire si on passait à 100, 1000 joueurs ? Ça deviendrait très vite fastidieux d'écrire 100 variables, de modifier la fonction moyenne pour prendre 100 paramètres,

notes

résumé

4m 19s

donc c'est simplement impossible pour un programmeur normal. Enfin, comment faire même si le nombre de joueurs n'est pas connu au départ ? Vous ne sauriez pas si vous devez mettre cinq, deux, ou dix joueurs. La réponse à ces nouveaux besoins est la même, c'est l'introduction d'un nouveau type, d'un type tableau.

notes

résumé

4m 30s

Solution avec tableau

```
unsigned int nb_joueurs(5);
vector<int, nb_joueurs> scores;

for(auto score : scores) {
  score = calcule_score(....);
}

// Calcul de la moyenne
int moyenne_joueurs(moyenne(scores));

// Affichages
cout << "Score         Ecart Moyenne" << endl;
for(auto score : scores) {
  cout << score << "     " << score - moyenne_joueurs << endl;
}
```

Je vous montre, juste en guise de préambule la version avec un tableau du programme du score des joueurs, mais bien sûr nous allons détailler tout ceci

notes

résumé

4m 49s

Les tableaux

EPFL

Un **tableau** est une collection de valeurs *homogènes* (= **même type**)

Exemple : Tableau `scores` contenant 4 `int`

1000	1500	2490	6450
`scores[0]`	`scores[1]`	`scores[2]`	`scores[3]`

Utilisation des tableaux : *stockage* de *plusieurs* variables de *même* type

On pourra définir des tableaux d'`int`, de `double`, de `bool`, ...

dans la suite de cette séquence vidéo. Donc, on déclarerait un tableau de joueurs, ensuite on ferait ici une boucle, on retrouve cette notion de boucle pour calculer le score de chacun des joueurs, puis on calculerait ici la moyenne, cette moyenne pourrait prendre un tableau de scores, et enfin on aurait donc l'affichage du score final avec ici une boucle for qui parcourrait le tableau. Voyons donc ces différents éléments un par un.

notes

résumé

4m 56s

Les différentes sortes de tableaux

Il existe en général quatre sortes de tableaux :

		taille initiale connue *a priori* ?	
		non	oui
taille pouvant varier lors de l'utilisation du tableau ?	oui	1.	2.
	non	3.	4.

Commençons par la notion même de tableau. Un tableau, c'est une collection, un ensemble de valeurs qui ont toutes le même type. On parle de collection homogène, pour dire que les données stockées ont toutes le même type. Par exemple, le tableau des scores qui nous intéressait précédemment, si je prenais ici le cas de quatre scores, ça serait un tableau qui contiendrait quatre entiers. J'ai décidé que mes scores étaient entiers. Donc le tableau en question a tous les éléments de même type, le type int. Quand est-ce qu'on utilisera un tableau dans un programme ? Justement, lorsque l'on a besoin d'utiliser plusieurs variables de même type, comme par exemple les scores de notre exemple de jeu avec les joueurs. En C++, on peut faire des tableaux de n'importe quel type, des tableaux d'entiers, des tableaux de double, on peut faire des tableaux de n'importe quel type à disposition, des tableaux de bool, et même, une fois qu'on aura ce nouveau type tableau, des tableaux de tableaux. Il existe de façon générale quatre sortes de tableaux, en fonction de deux questions, est-ce qu'on connait avant d'utiliser le tableau, au moment où on écrit le programme, la taille du tableau à utiliser ? Et est-ce que cette taille, une fois connue, peut varier ou est fixée ? Examinons les différents cas tour à tour. Commençons par le cas où la taille n'est pas connue au moment où j'écris le programme, et où la taille peut varier lorsque le programme se déroule. C'est typiquement l'exemple de l'ensemble des âges des étudiants qui suivent ce cours. Au début je n'ai aucune idée de combien d'étudiants vont être présents à ce cours ? Puis ensuite, à un moment donné quand j'ai le tableau de tous les âges des étudiants à ce cours, on peut très bien avoir des étudiants

notes

résumé

5m 22s

Les différentes sortes de tableaux

Il existe en général quatre sortes de tableaux :

		taille initiale connue *a priori* ?	
		non	oui
taille pouvant varier lors de l'utilisation du tableau ?	oui	1.	2.
	non	3.	4.

qui se réinscrivent, et donc rajouter des âges, ou malheureusement, on peut aussi avoir des étudiants qui quittent le cours, et donc avoir des âges qui disparaissent. Donc on a ici le cas où la taille peut varier, et elle n'est pas connue au préalable. Illustrons maintenant l'extrême inverse, où on suppose que la taille est connue à priori, et qu'elle ne peut pas évoluer. Un exemple typique, si vous voulez faire un programme d'algèbre linéaire, est celui des vecteurs en 2D. Vous savez qu'un vecteur en 2D a deux coordonnés tout le temps, et donc ce sera tout le temps un tableau avec deux nombres représentant les coordonnées x et y. La taille est connue, à priori elle vaut deux, et la taille ne changera pas, elle vaudra toujours deux. Pour le cas où la taille n'est pas connue au départ, mais une fois fixée, la taille n'évolue pas, on pourrait prendre l'exemple d'un jeu, avec un nombre fixé de joueurs. On ne connait pas au début combien on va avoir de joueurs, s'il y a trois personnes qui vont participer au jeu ou dix personnes, mais une fois que le nombre de joueurs est fixé, on n'a plus le droit de le varier.

notes

résumé

Donc ici la taille n'est pas connue au départ, mais une fois qu'elle est connue, à ce moment là, elle ne peut plus évoluer pendant que le programme se déroule. Enfin le dernier cas, peut-être un peu plus rare, et plus difficile à illustrer, c'est le cas où on connait effectivement la taille à priori, au moment où on écrit le programme, mais où cette taille peut quand même varier. Ce serait typiquement si on voulait faire un programme

notes

résumé

8m 2s

Les différentes sortes de tableaux

Il existe en général quatre sortes de tableaux :

		taille initiale connue *a priori* ?	
		non	oui
taille pouvant varier lors de l'utilisation du tableau ?	oui	1.	2.
	non	3.	4.

qui stocke des statistiques pour par exemple, le nombre de cantons dans un pays, c'est quelque chose que l'on connait. Au départ on a 26 cantons, et puis peut-être un jour, il y a un canton de plus qui se créera, ou deux cantons vont fusionner, et donc le nombre diminuerait.

notes

résumé

8m 25s

Les tableaux en C++

EPFL

En C++, on utilise :

		taille initiale connue *a priori* ? non	taille initiale connue *a priori* ? oui
taille pouvant varier lors de l'utilisation du tableau ?	oui	`vector`	(`vector`)
	non	(`vector`)	`array` (C++11) tableaux « à la C »

C'est un cas un petit peu limite, et j'avoue que, effectivement, ce cas là est beaucoup plus rare. Deux remarques complémentaires, pour conclure sur les tableaux. Remarquez d'abord que si on sait faire des tableaux à taille non fixée, et qui peut varier, alors on peut évidemment avec ces tableaux faire tous les autres, ça c'est une première remarque. Et ensuite deuxième remarque, c'est que pratiquement aucun langage de programmation n'offre les quatre variantes de façon explicite.

notes

résumé

8m 42s

D'ailleurs en C++, on n'a que deux types de tableaux, on va avoir les tableaux dynamiques, représentés par le type vector, et les tableaux statiques, ou de taille fixe, qui sont depuis C++ 2011, représentés avec le type array, ou alors avec un ancien type qui est hérité de l'ancêtre du C++, qu'on appelle le C, donc les tableaux "à la C". donc les tableaux "à la C".

notes

résumé

9m 9s