Support de cours

Cours:

Initiation à la programmation (en C++)

Vidéo:

Puissance 4 - introduction

Concepts (extraits des sous-titres générés automatiquement) :

Élément de la ligne. Types nécessaires. Façon générale. Entités d'une énumération. Cas de notre exemple. Programme final. Éléments possibles. Pion de sa couleur. Façons possibles. Affichage de la grille. Tel cas. Dernière leçon. Seule fois. Pions de sa couleur. Aide du mot-clé.

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

vers la vidéo

EPFL

Puissance 4 : introduction

Initiation à la programmation (C++)

Vincent Lepetit, Jean-Cédric Chappelier et Jamila Sam

...

notes

résumé

0m 0s

Dans cette dernière leçon, nous allons voir comment aborder un projet

notes

résumé

0m 1s

Le jeu

```
 | | | | | | | |
 | | | | | | | |
 | | | |X| | | |
 | | |X|O|O|X|O|
 | | |O|X|X|X|O|
 | |O|O|X|X|O|X|
==1=2=3=4=5=6=7==

Joueur O :  entrez un numero de colonne
5

 | | | | | | | |
 | | | | | | | |
 | | | |X|O| | |
 | | |X|O|O|X|O|
 | | |O|X|X|X|O|
 | |O|O|X|X|O|X|
==1=2=3=4=5=6=7==

Le joueur O a gagne !
```

plus ambitieux que ceux que nous avons considérés jusqu'ici ; c'est-à-dire comment décomposer ce projet en sous problèmes, plus faciles à résoudre, jusqu'à ce que l'on sache écrire le programme final. Alors, pour illustrer nos propos nous avons choisi un jeu de « puissance 4 ». Vous connaissez sans doute les règles du « puissance 4 ». Ça se joue à deux joueurs, avec une grille de 7 colonnes ayant 6 cases chacune. Chaque joueur laisse tomber tour à tour un pion de sa couleur dans la colonne de son choix, jusqu'à ce que un joueur réussisse à aligner 4 pions de sa couleur ou que la grille soit complètement remplie.

notes

résumé

0m 5s

Tout d'abord une précision : nous n'allons pas développer une jolie interface graphique. Ce qui nous intéresse ici c'est que notre programme suive la mécanique du jeu et nous allons nous contenter d'un affichage de la grille et des pions avec des caractères, c'est-à-dire quelque chose qui ressemblerait à ceci. De façon générale, comment devez-vous procéder

notes

résumé

0m 48s

Développement

1. N'essayez pas d'écrire tout le programme en une seule fois !
   - Décomposez le problème en sous-problèmes pour développer le programme par étapes ;
   - À chaque étape, testez le code développé.

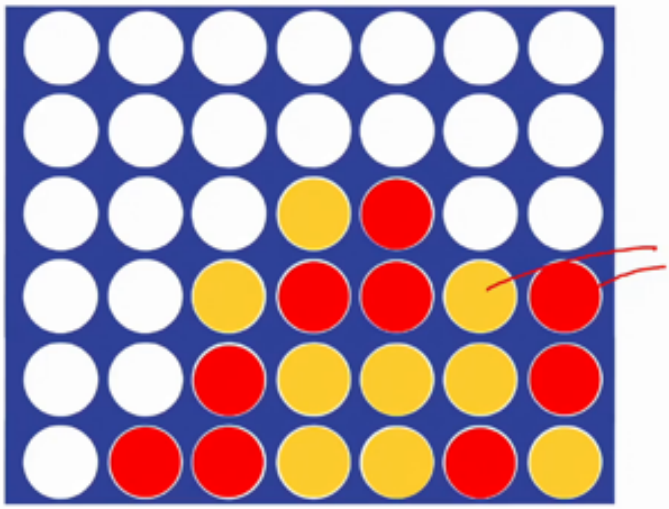
quand vous êtes face à un problème un peu conséquent tel que celui-ci ? Surtout il ne faut pas essayer d'écrire tout le programme en une seule fois ! Vous n'allez pas vous en sortir. Il faut savoir décomposer le problème en sous-problèmes pour écrire le programme par étapes. Et à chaque étape, il faut tester le code que vous aurez écrit pour continuer sur des bases solides.

notes

résumé

1m 13s

Identifier les types



Tout d'abord, il faut identifier les types nécessaires pour que le programme puisse fonctionner, c'est-à-dire, pour pouvoir représenter les données dont le programme a besoin. Dans le cas de notre exemple il s'agit essentiellement de pouvoir représenter la grille. Ensuite, il faudra identifier les fonctions qui portent sur ces types, écrire ces fonctions et les tester au fur et à mesure, et enfin quand une fonction est difficile à écrire, et bien on peut introduire une fonction pour chacun des points un peu difficiles. J'ai dit qu'il fallait commencer par identifier le type, alors qu'est-ce que ça veut dire et bien on va commencer par identifier les données dont a besoin le programme ; et trouver comment représenter ces données. En l'occurrence, pour notre programme de « puissance 4 », la donnée principale est la grille et les pions qu'elle contient. Plus exactement, on veut savoir pour chaque case ce que contient la case, Est-ce que c'est un pion rouge ou un pion jaune ou alors, rien du tout ?

notes

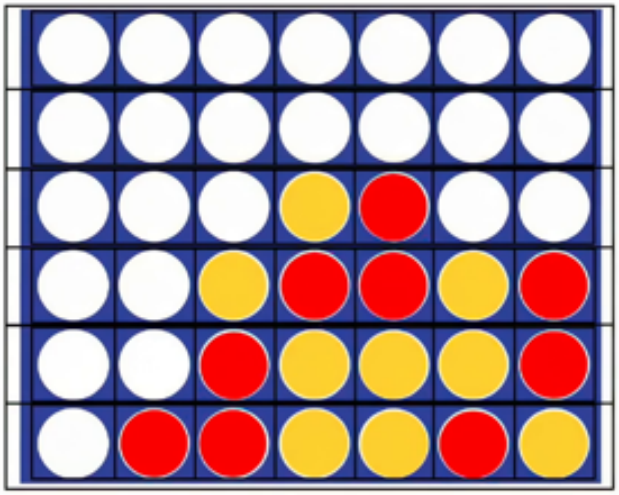
résumé

1m 33s

Identifier les types

EPFL

```
array<array<type_des_elements, 7>, 6>
```



Il y a plusieurs façons possibles de représenter la grille. Alors une façon de faire, c'est de voir la grille comme étant faite de lignes où chaque élément de la ligne correspond à une case. C'est-à-dire qu'on va représenter les lignes comme des tableaux. Alors il y a plusieurs possiblités pour représenter des tableaux en C++. Je rappelle qu'on peut utiliser soit le type « array », soit le type « vector ». Ici, comme on sait à l'avance le nombre de cases, plus exactement d'éléments possibles pour notre ligne, c'est-à-dire, on sait que chaque ligne aura 7 éléments, on va plutôt utiliser le type « array » pour représenter une ligne. Pour stocker l'ensemble des lignes on peut utiliser encore une fois un tableau dont chacun des éléments va contenir une des lignes de la grille, et comme le nombre de lignes est connu à l'avance, -- une grille de « puissance 4 » a toujours 6 lignes -- là encore, on va utiliser le type « array ». Le type qui va me permettre de représenter une grille de « puissance 4 » et ce qu'elle contient est donc un tableau de tableaux ou plus exactement un tableau à 6 éléments dont chacun des éléments est lui-même de type tableau

notes

résumé

2m 37s

Le type Grille

```
typedef array<array<type_des_elements, 7>, 6> Grille;

...

Grille grille;
```

à 7 éléments, et correspond à une ligne de la grille. Alors il me reste à définir le type des éléments d'une ligne, c'est-à-dire le type de ce que contient une case de la grille. Mais avant cela, j'aimerais faire une remarque : j'ai choisi de représenter la grille sous la forme d'un tableau de 6 lignes mais j'aurais pu faire autrement par exemple, j'aurais pu représenter la grille sous la forme d'un tableau de 7 colonnes. Alors de manière générale il y a plusieurs façons d'écrire un programme pour résoudre le même problème. On va vous donner une solution d'un programme de « puissance 4 », mais évidemment il y a plusieurs façons de faire.

notes

résumé

4m 1s

Le type Grille

```
typedef array<array<type_des_elements, 7>, 6> Grille;

...

Grille grille;
```

On va utiliser le mot clé « typedef » pour définir un nouveau type, « Grille », comme étant un tableau de tableaux de la bonne taille c'est-à-dire de la taille

notes

résumé

4m 41s

Type des éléments

```
typedef array<array<type_des_elements, 7>, 6> Grille;
type_des_elements ?
```

- `int` ? avec par exemple 0 pour vide, 1 pour rouge, 2 pour jaune ;
- `char` ? avec par exemple `' '` pour vide, `'O'` pour jaune, `'X'` pour rouge.

d'une grille de jeu de « puissance 4 ». Et ce nouveau type va nous permettre de définir des variables qui seront donc des tableaux de tableaux permettant de représenter une grille de « puissance 4 ». La question qui se pose maintenant c'est : comment va-t'on pouvoir représenter les éléments de la grille ? On pourrait par exemple utiliser le type « int » et décider d'une convention qui dirait que si un élément contient la valeur '0' et bien cet élément correspond à une case vide ; s'il contient '1'

notes

résumé

4m 51s

Type énuméré avec `enum` (1)

EPFL

En C++, il est possible de donner des noms aux entités d'une énumération, comme par exemple les couleurs possibles des pions, ou la liste des pays d'un continent, etc.

ça correspond à une case occupée par un pion rouge ; s'il contient 2, à une case occupée par un pion jaune. Ou alors on pourrait utiliser des caractères avec le type « char » et décider que par exemple un espace correspond à une case vide ; le caractère 'O' correspondrait à une case occupée par un pion jaune ; le caractère 'X' à une case occupée par un pion rouge. Mais ces deux solutions ont un inconvénient. Si par exemple on choisit le type « int », rien n'empêche qu'un élément de la grille contienne la valeur 3 par exemple alors que 3 ne correspond à rien pour notre programme. Même si l'on choisit le « char », un élément de la grille pourrait contenir le caractère 'z' qui également ne correspond à rien pour notre jeu. Dans un tel cas, il faut définir un nouveau type à l'aide du mot-clé « enum ». Il va nous permettre de limiter le nombre de valeurs possibles que vont prendre les éléments de notre grille. En effet, en C++ on peut donner des noms aux entités d'une énumération, où une énumération

notes

résumé

5m 25s

Type énuméré avec enum (2)

EPFL

```
enum Couleur { vide, rouge, jaune };
```

On peut alors utiliser ces valeurs comme pour un type entier, par exemple:

```
Couleur element(vide);

element = jaune;
// ...
if (element == rouge) {
  // ...
}
```

est par exemple l'ensemble des couleurs possibles pour un pion, ou alors la liste des pays d'un continent. Ça va se faire en utilisant le mot-clé « enum » en respectant la syntaxe suivante. Tout d'abord donc le mot-clé « enum », suivi du nom qu'on veut donner à l'énumération, suivi, entre accolades ouvrantes et fermantes, de la liste des noms qu'on veut donner aux valeurs de l'énumération, et chacun des noms devra être séparé par une virgule (','). L'instruction se termine par un point-virgule (';'). Et dans notre programme nous allons utiliser l'énumération définie de la façon suivante : « enum », suivi du nom de l'énumération. Dans notre cas, cette énumération va servir à définir la valeur des éléments d'une grille ; donc une bonne idée ce serait par exemple d'appeler cette énumération « Couleur ». Ensuite, ces éléments vont correspondre soit à une case vide, soit à une case qui va contenir un pion de couleur rouge, soit une case qui va contenir un pion de couleur jaune. Donc pour les noms des entités de l'énumération nous allons utiliser simplement « vide », « rouge » et « jaune ». Une fois cette énumération définie, on peut l'utiliser comme n'importe quel autre type pour déclarer des variables. Cette instruction-ci par exemple

notes

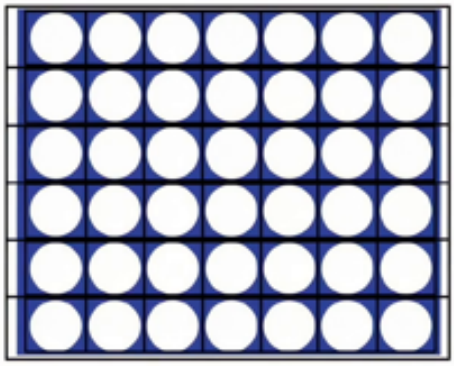
résumé

6m 37s

Types de notre programme

EPFL

```
enum Couleur { vide, rouge, jaune };
```



déclare une variable qui s'appelle « element », qui sera donc de type « Couleur » et qui sera initialisée à la valeur « vide ». On peut également changer la valeur de l'élément à l'aide d'une affectation, comme dans cette instruction qui va changer « vide » en « jaune », ici. Et on peut également tester ce que contient la variable élément avec cette condition, est-ce que cette variable élément contient la valeur « rouge » ? Dans ce cas évidemment, le test est faux.

notes

résumé

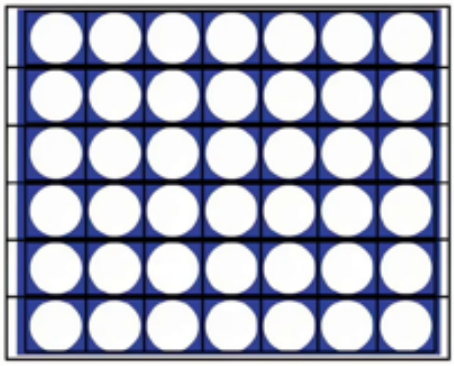
8m 1s

Types de notre programme

EPFL

```
enum Couleur { vide, rouge, jaune };

typedef array<array<Couleur, 7>, 6
```



Les types de notre programme vont donc être notre énumération « Couleur », ainsi que le type « Grille » qui va être finalement défini de la façon suivante.

notes

résumé

8m 31s

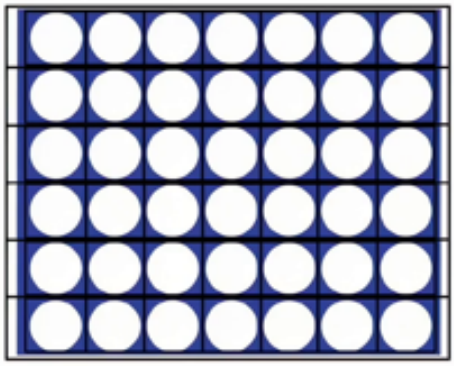
Types de notre programme

```
enum Couleur { vide, rouge, jaune };

typedef array<array<Couleur, 7>, 6> Grille;

...

Grille grille;
grille[2][3] = jaune;
```



Vous pourrez remarquer que j'ai utilisé ici le type « Couleur » pour définir le type des éléments. Une fois défini ce type « Grille », je vais pouvoir m'en servir pour déclarer des variables. Comme cette variable que je vais appeler « grille ». Cette variable, qui est donc un tableau, je vais pouvoir la modifier en utilisant des instructions qui vont ressembler à celles-ci. Cette instruction affecte la valeur « jaune » à l'élément d'indice 3 de la ligne indice 2 du tableau « grille » Encore faut-il savoir à quel élément de la vraie grille nous faisons référence dans cette affectation.

notes

résumé

8m 41s

Alors, pour cela, il faut faire un choix. Est-ce que la ligne d'indice 0 de notre tableau « grille » correspond à la ligne du haut de la vraie grille ? Ou est-ce que la ligne d'indice 0 correspond au bas de la vraie grille ? Alors il n'y a pas de bonne façon de décider, il n'y a pas de façon qui est meilleure qu'une autre, et on va décider arbitrairement dans notre cas de dire que la ligne 0 correspond à la ligne du haut de la grille. on peut donc savoir maintenant quel est le résultat de cette affectation pour notre vraie grille. A considérer la ligne 0, 1, 2 et l'élément d'indice 3 sur cette ligne. Cet élément è l'indice 0 celui-ci a l'indice 1, 2, 3. Et donc, notre affectation va modifier la valeur de cet élément et va lui mettre la valeur « jaune » pour définir que cette case contient un pion jaune. Dans la vidéo suivante, nous allons voir comment écrire des fonctions qui vont nous permettre d'initialiser et d'afficher la grille. et d'afficher la grille.

notes

résumé

9m 25s