Support de cours

Cours:

Initiation à la programmation (en C++)

Vidéo:

Blocs d’instructions (partie 4)

Concepts (extraits des sous-titres générés automatiquement) :

Variable i. Nom de variable. Règles de résolution de portée. Cas particulier de portée. Boucle itérative. Boucle for. Portée des variables. Partie condition. Langages évolués. Instruction for. Partie incrémentation du for. Programme complet. Choses différentes. Fin du programme. Programmes.

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

vers la vidéo

EPFL

Fonctions : blocs
(Partie 4)

Initiation à la programmation (C++)

Vincent Lepetit, Jean-Cédric Chappelier et Jamila Sam

...

notes

résumé

0m 0s

Règles de résolution de portée

```
if (i != 0) {
  int j(0);

  ...
  j = 2 * i;
  ...
  if (j != 2) {
    int j(0);

    ...
    j = 3 * i;
    ...
  }
  ...
}
```

En cas d'ambiguïté, c'est-à-dire quand plusieurs variables de portées différentes portent le même nom:

la variable « la plus proche » est choisie

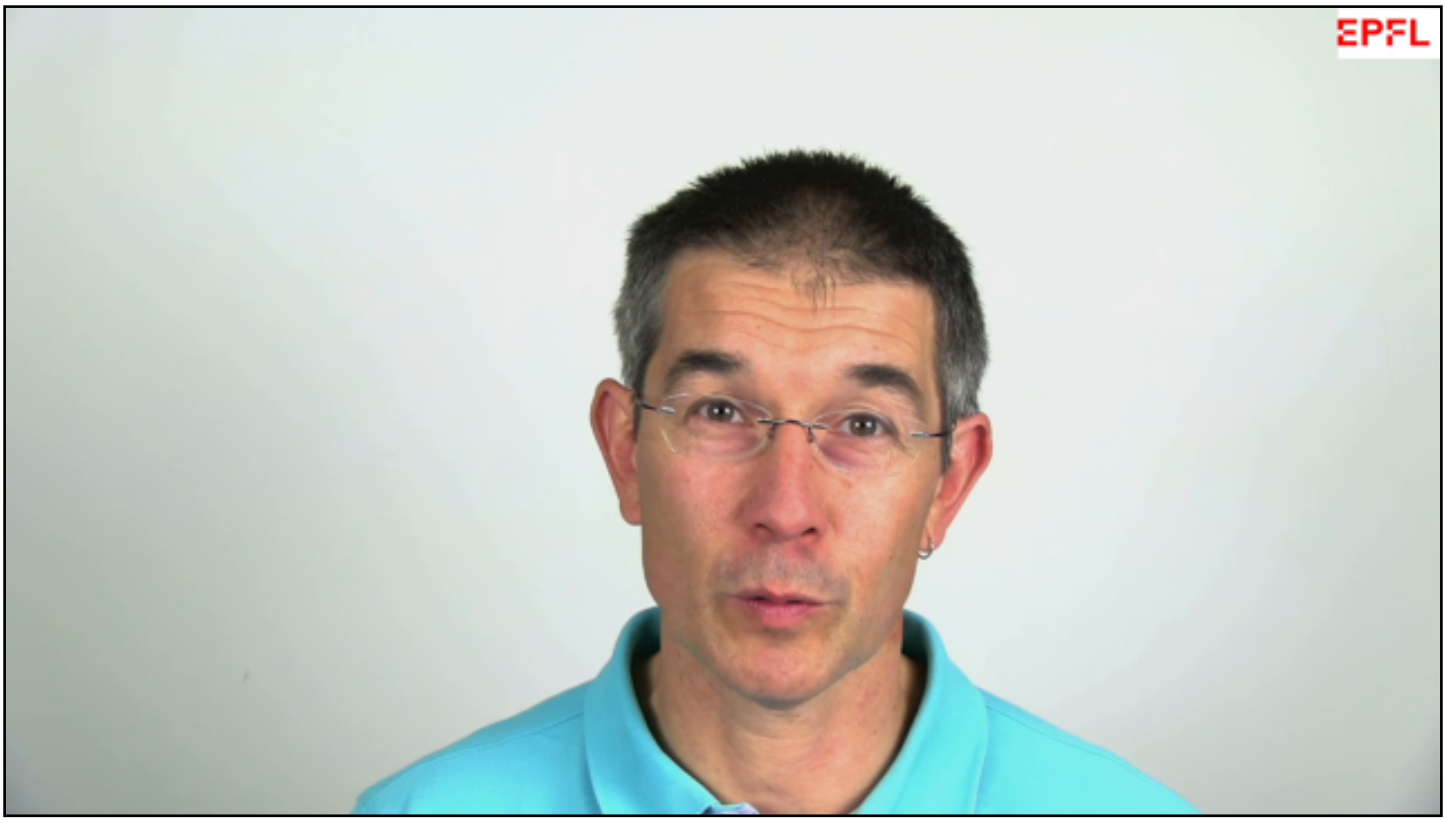
Bonnes pratiques:
Évitez d'utiliser plusieurs fois le même nom de variables, sauf peut-être pour des variables locales aux boucles, comme `i`, `j`, ...

Ceci donc pour illustrer les règles de résolution de portée en C++. Mais peut-être ce qu'il faut avant tout retenir de cette explication c'est

notes

résumé

0m 1s



qu'il ne faut jamais utiliser plusieurs fois le même nom de variable. Evitez toute ambiguïté. J'ai l'habitude de dire qu'on n'écrit pas des programmes pour les machines, sinon on continuerait à écrire en assembleur comme au tout début, mais on écrit dans des langages évolués avant tout pour les humains, pour écrire des programmes qui sont compréhensibles, maintenables, que l'on peut corriger et transmettre vers des humains et donc dans cet esprit-là,

notes

résumé

0m 13s

Portée : cas des itérations

La déclaration d'une variable à l'intérieur d'une itération est une déclaration **locale au bloc de la boucle**, et aux deux instructions de test et d'incrément:

```
for(int i(0); i < 5; ++i) {
  cout << i << endl;
}
// A partir d'ici, on ne peut plus utiliser ce i
```

évitez toute ambiguïté dans vos programmes et essayez d'être au maximum le plus clair possible et donc ne nommez pas des choses différentes avec le même nom. Un cas particulier de portée qu'il faut connaître, c'est la portée des variables qui sont déclarées dans une boucle itérative, dans un for. Ici dans la boucle for, nous avons déclaré une variable i. La portée de ce i va donc être le bloc contrôlé par l'instruction for, ainsi bien sûr que la partie condition et la partie incrémentation du for, mais la portée de ce i n'ira pas plus loin donc que

notes

résumé

0m 37s

Exemple (à ne pas suivre !)

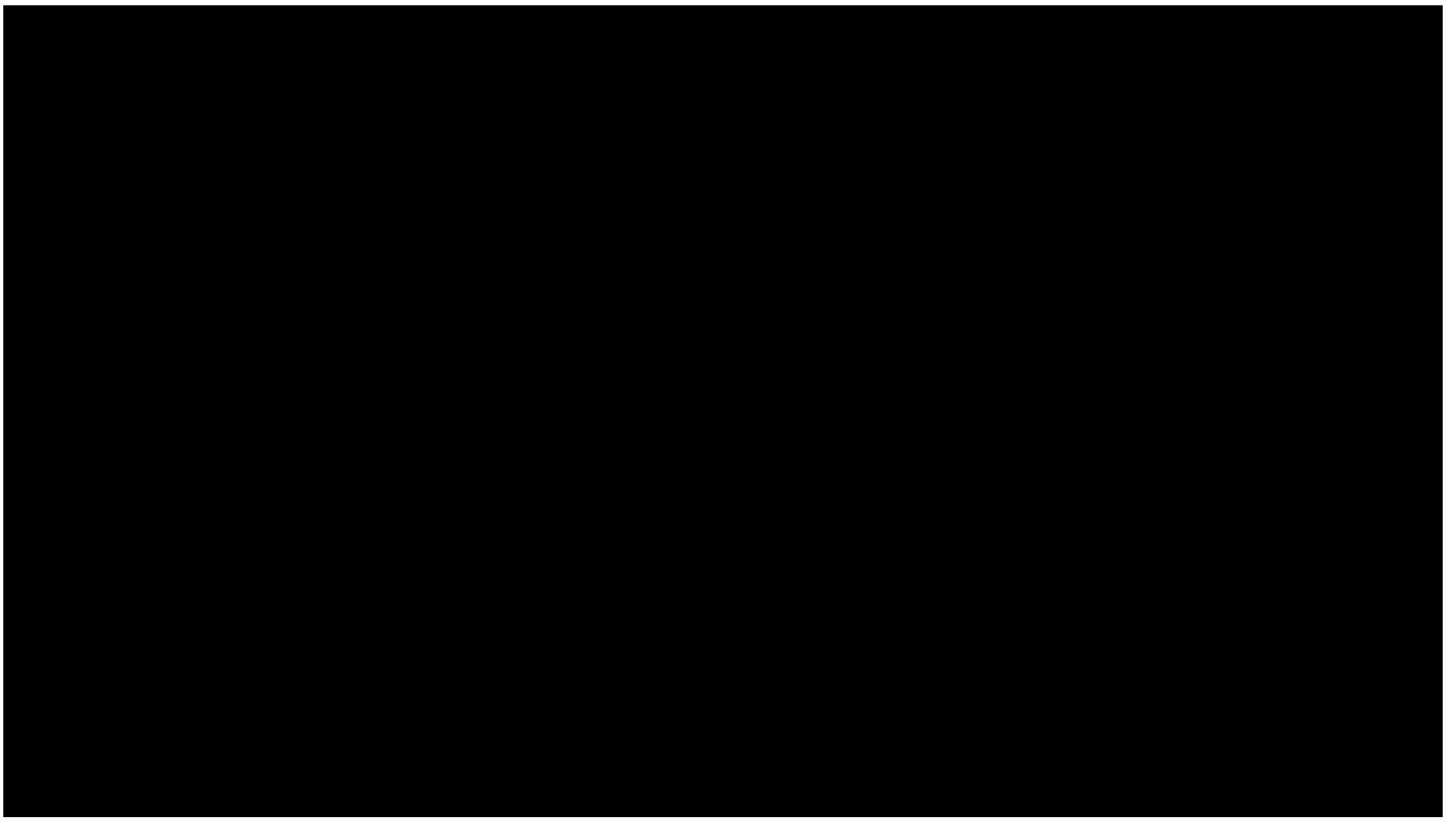
```
#include <iostream>
using namespace std;

int main()
{
  int i(120);

  for(int i(0); i < 5; ++i) {
    cout << i << endl;
  }

  cout << i << endl;

  return 0;
}
```

ce for, à partir de cet endroit-là on ne peut plus utiliser cette variable i, cette variable i est locale à la boucle for. Terminons donc maintenant par une exemple à ne pas suivre si vous suivez le conseil de ne pas nommer avec le même nom deux choses qui sont différentes, mais juste pour voir si vous avez bien compris les règles de résolution de portée. Donc ici on a un programme complet, dans le

notes

résumé

1m 13s



main on déclare ici une variable i de type entier que l'on initialise à la valeur cent vingt et puis on a ensuite une boucle for, ici, qui va itérer pour un i qui va de zéro jusqu'à cinq et on affiche donc la valeur de i dans la boucle, on affiche la valeur de i ici, à la fin du programme. La question c'est qu'affiche le programme? La question c'est qu'affiche le programme?

notes

résumé

1m 37s