Support de cours

Cours:

Initiation à la programmation (en C++)

Vidéo:

Variables (partie 4)

Concepts (extraits des sous-titres générés automatiquement) :

Égalité mathématique. Deuxième situation. Tournure de cette nature. Symbole égal. Résultat de l'évaluation. Première instruction. Droite de l'opérateur d'affectation. Long des calculs. Nouvelle valeur. Instruction suivante. Valeurs différentes. Travers des exercices. Variables. Propre valeur. Instructions.

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

vers la vidéo

EPFL

Variables
(Partie 4)

Initiation à la programmation (C++)

Vincent Lepetit, Jean-Cédric Chappelier et Jamila Sam

...

notes

résumé

0m 0s

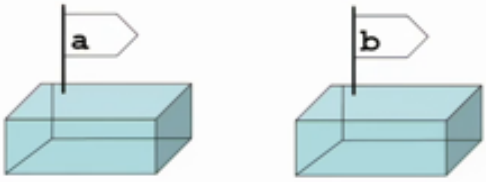
En mathématiques:

$b = a + 1$

signifie que tout au long des calculs, a et b vérifieront toujours cette relation. Autrement dit, quel que soit a, b sera toujours égal à $a+1$

En C++:

```
a = 5;
b = a + 1;
a = 2;
```



Comme je l'ai dit tout à l'heure, il ne faut pas confondre l'affectation avec une égalité mathématique. Le symbole égal est le même, mais en mathématique et en programmation, il ne signifie pas la même chose. Si on s'intéresse ici à ces deux instructions, ces deux lignes en mathématique signifieraient que a et b ont les mêmes valeurs dans les deux cas. En c++, ce n'est pas exactement ce qui va se passer. Intéressons-nous à la première instruction. Imaginons que nous ayons des variables a et b, stockant chacune deux valeurs différentes. Exécuter cette instruction aura pour conséquence d'évaluer le b, qui a pour valeur deux, de prendre le résultat de l'évaluation de b, et de le stocker dans a, ce qui produira ce résultat. Si on s'intéresse à la deuxième situation, imaginons que l'on ait à nouveau les variables a et b, avec des valeurs un et deux. Je vais évaluer a et prendre le résultat de l'évaluation de a pour le stocker dans b. Je prends a, je le mets dans b. Donc je me retrouve avec un a et un b qui ont tous les deux la valeur une. On voit bien qu'en programmation, a et b se retrouvent certes avec les mêmes valeurs, mais ce ne sont pas les mêmes valeurs dans les deux cas. Qui plus est, si plus tard l'un des deux est modifié, l'autre ne l'est pas. Par exemple, si a est ensuite modifié en trois, b garde sa propre valeur un. De même si en mathématique, j'écris quelque chose comme ceci, je signifie que tout au long des calculs,

notes

résumé

0m 1s

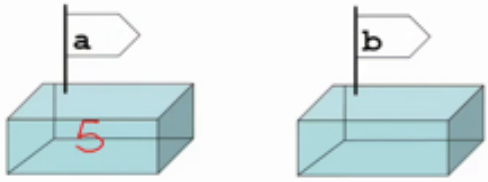
En mathématiques:

$b = a + 1$

signifie que tout au long des calculs, a et b vérifieront toujours cette relation.
Autrement dit, quel que soit a, b sera toujours égal à $a+1$

En C++:

```
a = 5;
b = a + 1;
a = 2;
```

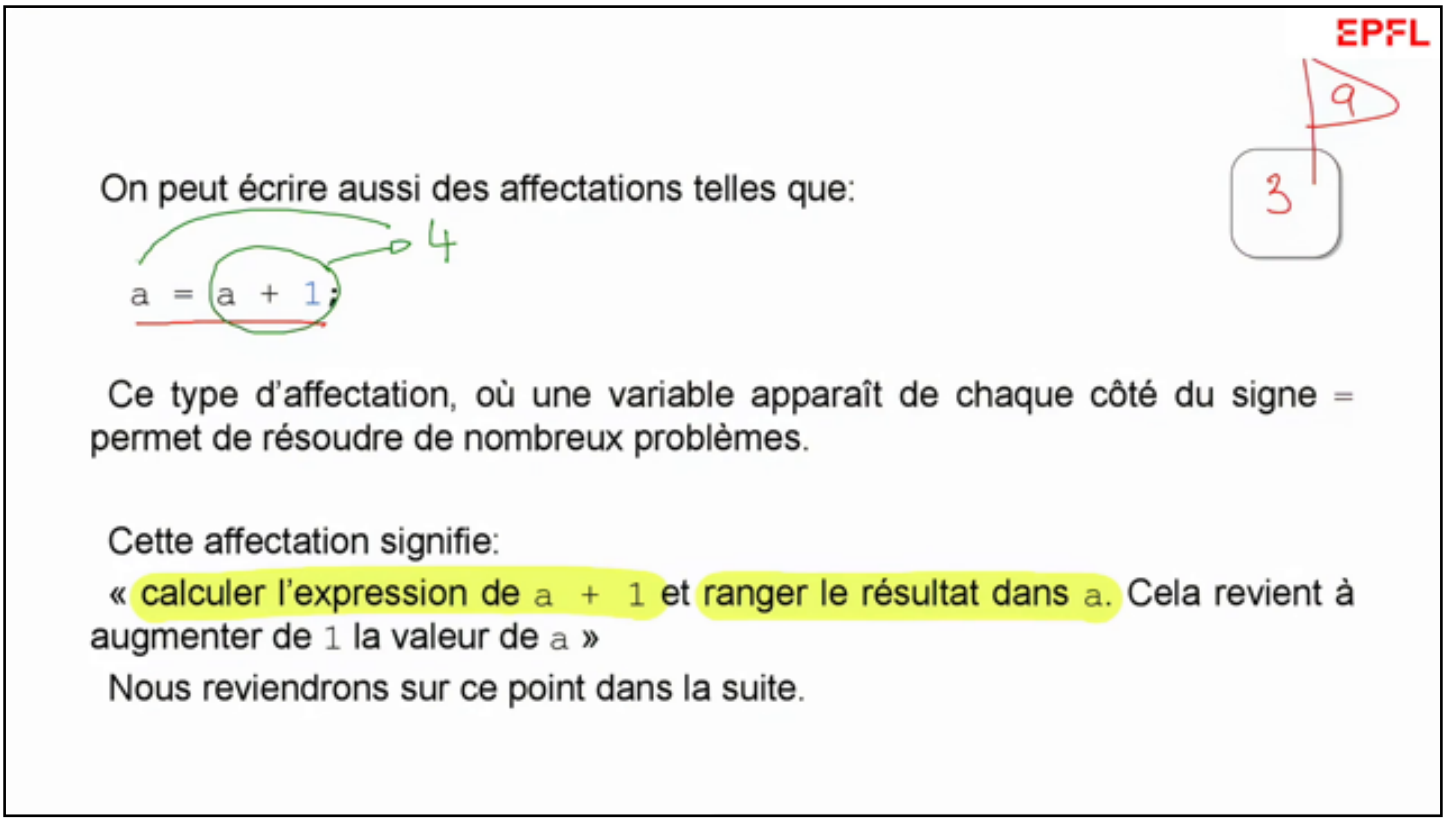


a et b vérifieront toujours cette relation. En c++, la situation est différente. Ici, imaginons que nous ayons la situation où j'affecte à la variable a la valeur cinq. Ensuite, j'écris une ligne qui ressemble très fortement à ce que j'exprimais ici en mathématique, et qui va avoir pour conséquence,

notes

résumé

1m 37s



de mettre dans b, le résultat de l'évaluation de a plus un, qui ici serait six. Ensuite, je peux parfaitement, lorsque j'exécute l'instruction suivante, changer à nouveau la valeur de a. On voit bien qu'ici, cette relation n'est plus vérifiée, comme elle le serait en mathématique. Je peux donc en c++ parfaitement écrire une tournure de cette nature, qui va signifier, calculer l'expression de a plus un, puis ranger le résultat dans a. Ici, très concrètement, si on a une variable a, qui contenait à l'origine par exemple, trois, je vais commencer par évaluer l'expression qui est à droite de l'opérateur d'affectation, ce qui va me produire le résultat, trois plus un, donc quatre.

notes

résumé

2m 1s

Et ensuite seulement, ranger la nouvelle valeur ainsi calculée dans a, ce qui produira ce résultat. Il s'agit là d'une tournure très fréquente, et très utile en programmation, que vous aurez l'occasion de pratiquer de façon très intensive, au travers des exercices. au travers des exercices.

notes

résumé

2m 49s