Support de cours

Cours:

Initiation à la programmation (en C++)

Vidéo:

Conditions (partie 1)

Concepts (extraits des sous-titres générés automatiquement) :

Valeur d'une variable. Compilateurs c. Valeur d'une variable n. Genre de tournure. Petit détail syntaxique. Conditions simples. Opérateurs de comparaison. Instruction de branchement conditionnel. Sorte d'expression. Termes de bonnes pratiques. Séquence vidéo précédente. Petite mise. Variable n. Symboles égal. Condition simple.

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

vers la vidéo

EPFL

Conditions
(Partie 1)

Initiation à la programmation (C++)

Vincent Lepetit, Jean-Cédric Chappelier et Jamila Sam

...

notes

résumé

0m 0s

Dans une séquence vidéo précédente, nous avons

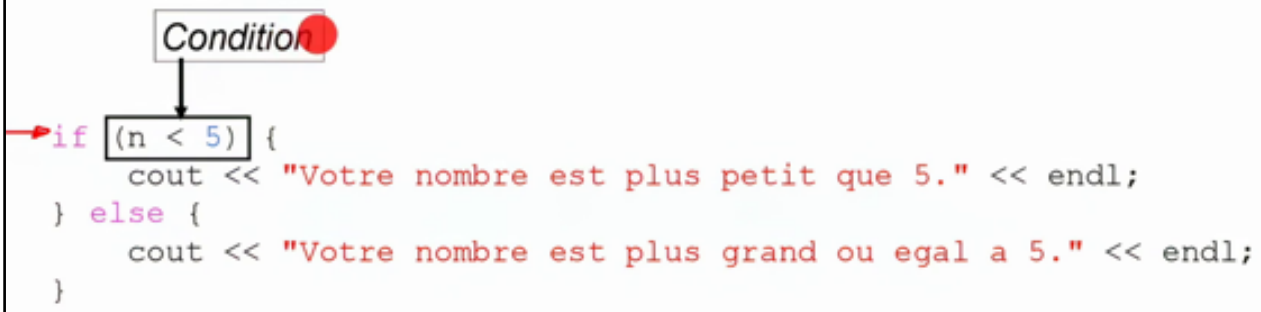
notes

résumé

0m 1s

Les conditions

L'instruction `if` fait apparaître une **condition** entre parenthèses



Attention, la condition est toujours entourée de parenthèses.

introduit l'instruction de branchement conditionnel, le if. Et nous avons vu que, pour pouvoir fonctionner, cette instruction a besoin d'exprimer des conditions. Les conditions que nous avons employées jusqu'ici étaient des conditions simples, qui consistaient par exemple à comparer la valeur d'une variable avec une valeur. Nous allons voir maintenant, qu'en C++, il est possible d'exprimer des conditions plus complexes. Vous avez ici un exemple d'instruction de branchement conditionnel, un if tel que nous avons déjà eu l'occasion d'en rencontrer précédemment. Nous voyons que pour écrire cette instruction, nous avons besoin

notes

résumé

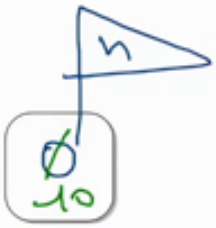
0m 5s

Valeurs des conditions

Une condition est un cas particulier d'expression, qui ne peut prendre que deux valeurs.

En C++, ces valeurs se notent `true` et `false`.

- Une condition vaut `true` quand elle est vraie, et
- une condition vaut `false` quand elle est fausse.



(n < 5) vraie ⟶ true

(n < 5) faux ⟶ false

de formuler une condition, ce que l'on voit concrètement ici. La condition dans le cas présent, est une condition simple, qui consiste à comparer la valeur d'une variable n avec la valeur 5. Petit détail syntaxique: la condition dans une instruction de branchement conditionnel est toujours entourée de parenthèses, ce que l'on voit concrètement ici. Une condition en C++, est en fait une sorte d'expression dont la particularité est de retourner deux valeurs possibles. Ces valeurs sont soit "true", soit "false". Tout simplement, une condition sera évaluée à "true" quand elle est vraie, et évaluée à "false" lorsqu'elle est fausse. Prenons un exemple concret, imaginons que j'ai une variable n dont la valeur est zéro, et j'écris la condition n inférieur à 5 que je veux évaluer. Ici la valeur n, zéro, est bien inférieur à 5, ce qui veut dire que cette condition est vraie, lorsque je l'évalue elle va donc retourner la valeur "true". Imaginons maintenant que n vaille 10. J'évalue la même expression. Evidemment ceci n'est plus vérifié, la condition est fausse, par conséquent l'évaluation de la condition va me retourner la valeur "false".

notes

résumé

0m 37s

Les conditions simples
Les opérateurs de comparaison

EPFL

Une **condition simple** compare deux expressions.

Elle utilise un **opérateur de comparaison**, comme < ou >

Opérateurs de comparaison du langage C++:

Opérateur de comparaison	Signification
<	inférieur à
>	supérieur à
==	égal à
<=	inférieur ou égal à
>=	supérieur ou égal à
!=	différent de

Commençons par examiner de façon un peu plus exhaustive

notes

résumé

1m 49s

Les conditions simples
Les opérateurs de comparaison

EPFL

Une **condition simple** compare deux expressions.
Elle utilise un **opérateur de comparaison**, comme < ou >

Opérateurs de comparaison du langage C++:

Opérateur de comparaison	Signification
<	inférieur à
>	supérieur à
==	égal à
<=	inférieur ou égal à
>=	supérieur ou égal à
!=	différent de

(x < y)

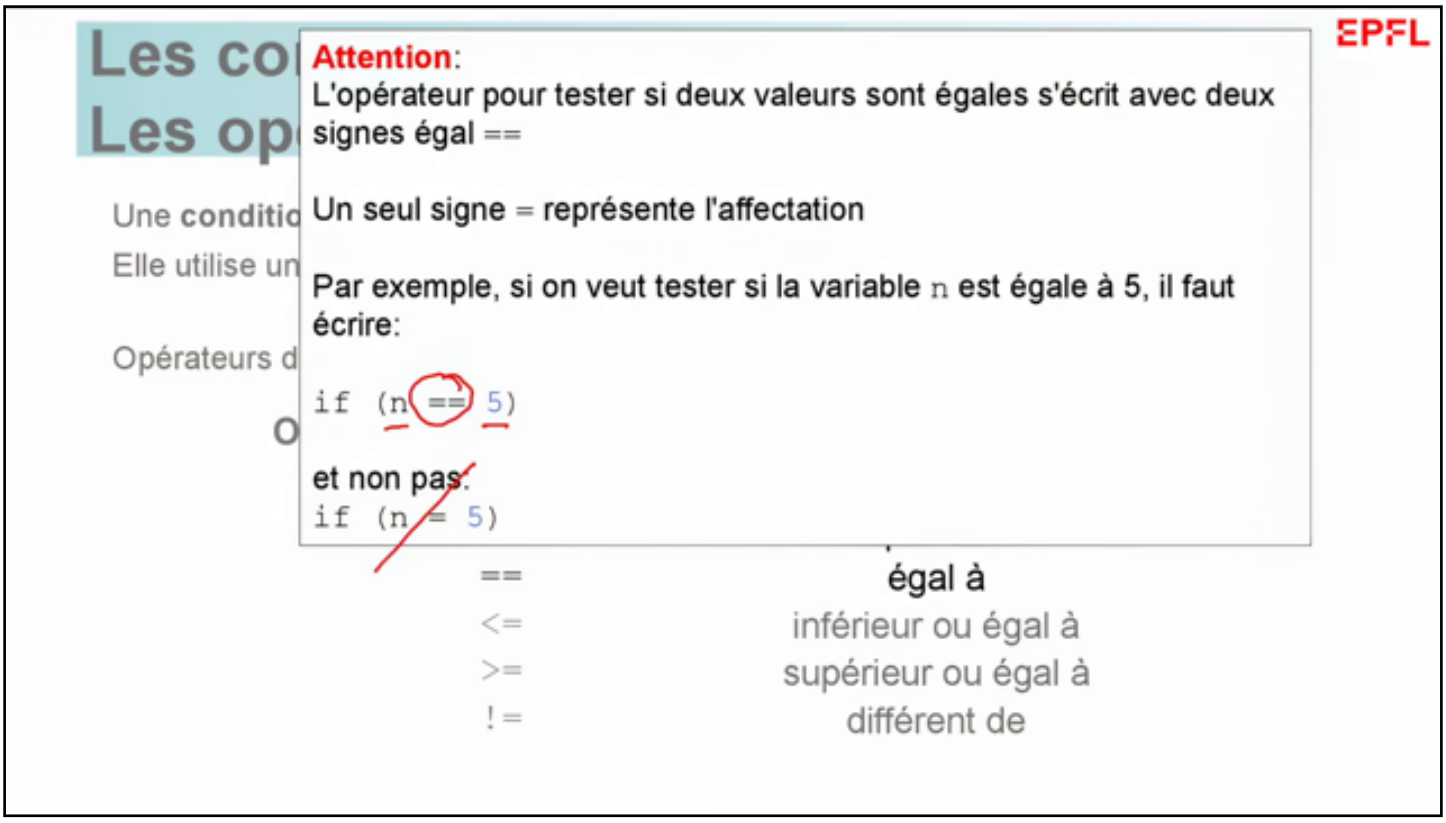
x = y

comment on peut formuler des conditions simples, en C++. La vocation d'une condition simple est de comparer deux expressions, et pour cela nous avons besoin d'utiliser des opérateurs de comparaison. Inférieur, supérieur en sont deux exemples que vous avez déjà eu l'occasion de croiser précédemment. Donc évidemment C++ offre toute une palette d'opérateurs prédéfinis pour formuler des conditions simples, imaginons par exemple que j'ai deux variables x, y, je veux savoir si la valeur de x est inférieure à la valeur de y, naturellement je vais employer l'opérateur de comparaison inférieur et écrire ma condition de cette façon. De même, si je veux savoir si la valeur de x est la même que la valeur de y, je vais utiliser l'opérateur de comparaison ==,

notes

résumé

1m 51s

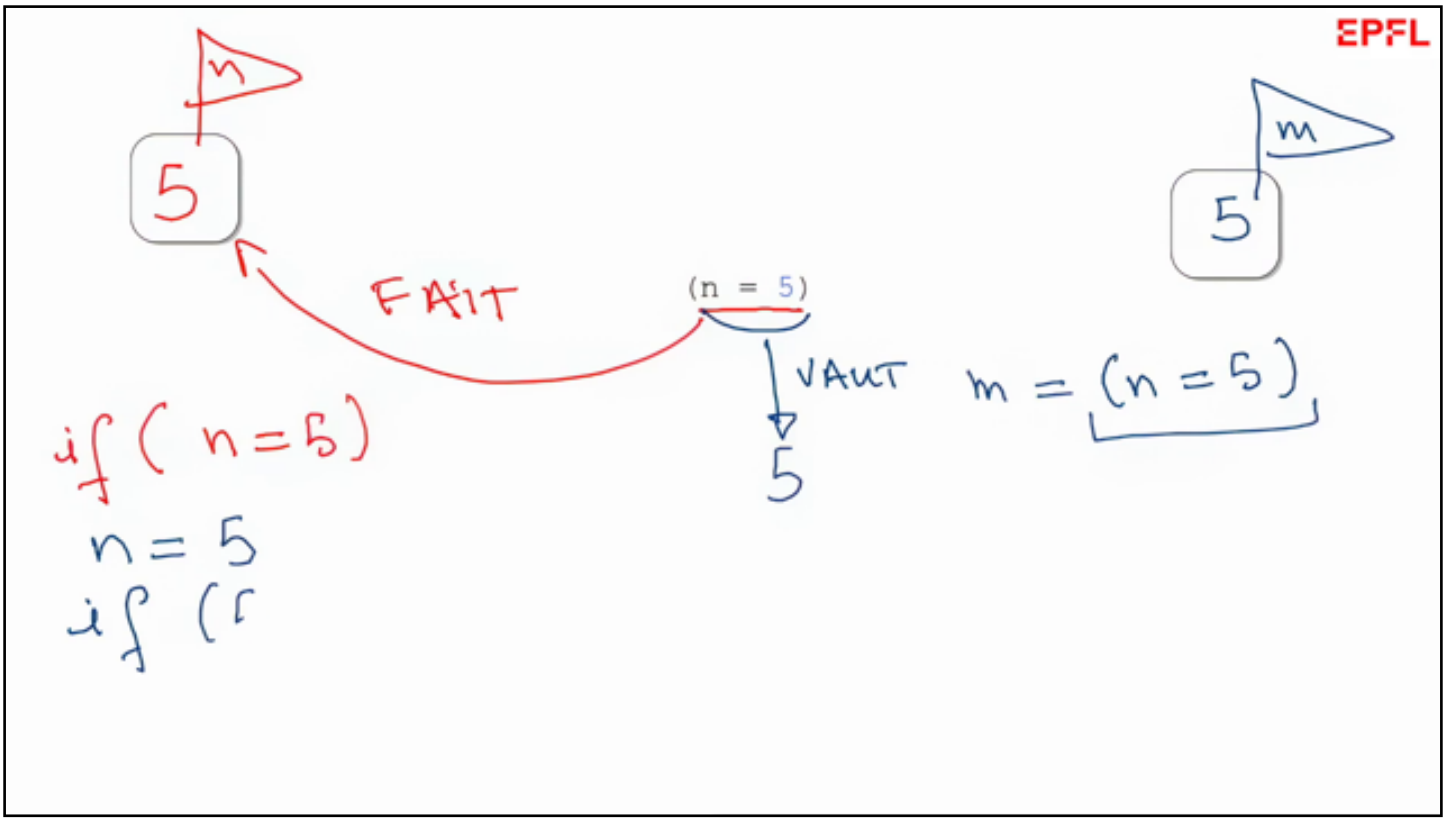


qui me permet de tester cette condition. Vous disposez des tenants inférieur ou égal, supérieur ou égal, et différent, qui permettent d'exprimer toutes sortes de conditions simples. Petite mise en garde de rigueur: l'opérateur utilisé en C++ pour savoir si deux valeurs sont égales, se formule au moyen de deux symboles égal. À ne pas confondre donc avec le symbole = tout court, qui lui est utilisé pour formuler une affectation. Si je veux savoir si la valeur d'une variable n est 5, je vais donc utiliser l'opérateur == pour formuler ma condition, et non pas l'opérateur égal tout court.

notes

résumé

2m 37s



Ce qu'il faut savoir cependant, c'est que les compilateurs C++ vont considérer ce genre de tournure comme licite, nous allons examiner pourquoi dans le transparent suivant, licite, certes, mais néanmoins déconseillée, en tout cas à ce stade d'apprentissage du langage. Ce qu'il faut savoir c'est qu'en C++, toute expression, quelle qu'elle soit, fait quelque chose, mais vaut aussi quelque chose. Examinons ceci sur un cas concret. Nous avons ici une expression, n = 5, Cette expression fait clairement quelque chose. Ce qu'elle fait, c'est affecter à une variable n la valeur 5. En C++, cette expression vaut aussi quelque chose. Ce qu'elle vaut, c'est la valeur de n après l'affectation. Donc l'expression n = 5 vaut 5. C'est pourquoi il est parfaitement licite d'écrire quelque chose comme ceci: m = n = 5, ce qui a pour vocation d'affecter à une variable m ce que vaut cette expression, c'est-à-dire 5. Il est donc tout à fait correct d'écrire quelque chose comme ceci: "if (n = 5)", ce qui est équivalent à écrire "if (5)", sachant qu'en C++ toute valeur non nulle est équivalente à "true", cette condition sera toujours évaluée comme vérifiée. Notez qu'en termes de bonnes pratiques, il est tout à fait déconseillé d'avoir recours à ce genre de tournure, je ne l'ai montrée ici de façon détaillée et explicite que parce que confondre == et = pour effectuer une comparaison est une erreur couramment commise par les débutants. Donc dans l'absolu, ce genre de tournures est à éviter, même si elles sont tolérées par le compilateur. compilateur.

notes

résumé

3m 13s