Support de cours

Cours:

Initiation à la programmation (en C++)

Vidéo:

Erreurs de débutant, le type bool (partie 1)

Concepts (extraits des sous-titres générés automatiquement) :

Message d'avertissement. Premier piège. Branchements conditionnels. Programmeurs débutants. Troisième erreur possible. Bonne idée. Instruction cout. Nouveau code. Intérieur du branchement conditionnel. Plupart des compilateurs. Erreur de syntaxe. Mot-clé else. Message d'erreur. Façon suivante. Seul symbole.

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

vers la vidéo

EPFL

Erreurs de débutant
Le type `bool`
(Partie 1)

Initiation à la programmation (C++)

Vincent Lepetit, Jean-Cédric Chappelier et Jamila Sam

...

notes

résumé

0m 0s

Voyons maintenant, plusieurs erreurs commises fréquemment par des

notes

résumé

0m 1s

Erreurs classiques

Le test d'égalité s'écrit ==, et pas =

programmeurs débutants, quand il s'agit d'écrire des branchements conditionnels. Le premier piège qui vous guette, est un piège dans lequel sont tombés probablement tous les programmeurs débutants, voire moins débutants, est que le test d'égalité s'écrit avec deux symboles égal, et non pas un seul symbole, qui est pour l'affectation.

notes

résumé

0m 5s

Erreurs classiques

Le test d'égalité s'écrit ==, et pas =

affectation

Par exemple, ce code-ci

notes

résumé

0m 25s

Erreurs classiques

Le test d'égalité s'écrit ==, et pas =

```
if (a = 1) // !!!
```

est accepté par le compilateur mais

- ne teste pas si `a` vaut 1, et
- affecte la valeur 1 à `a`.

Utilisé avec `-Wall`, `g++` affiche le *warning* suivant:

```
warning: suggest parentheses around assignment used as truth value
```

ou si vous avez une installation en Français:

```
attention : parenthèses suggérées autour de l'affectation utilisée comme valeur de vérité
```

est accepté par le compilateur, mais ne va pas tester si a est égal à un. Pour cela, il aurait fallu écrire if a égal, égal un. Mais en plus, il va affecter la valeur un, à la variable a. Alors, même si le compilateur accepte ce code, la plupart des compilateurs vont afficher un warning. C'est-à-dire, un message d'avertissement.

notes

résumé

0m 33s

Erreurs classiques

```
if (a == 1); // !!!
  cout << "a vaut 1" << endl;
```

Par exemple, g plus plus, avec l'option moins Wall, va afficher ce message-ci, ou ce message-ci, si vous avez une version en français. Donc, si vous obtenez un de ces deux messages, vous savez que vous aussi, vous êtes tombés dans ce piège. Peut-être, voyez-vous l'erreur dans ce nouveau code, qu'il ne devrait pas y avoir deux points-virgules, ici.

notes

résumé

1m 1s

Erreurs classiques

```
if (a == 1); // !!!
  cout << "a vaut 1" << endl;
```

`a vaut 1` est toujours affiché quelle que soit la valeur de `a`!

Le point-virgule est considéré comme une instruction, qui ne fait rien.

Le code précédent est compris par le compilateur comme:

```
if (a == 1)
  ;
cout << "a vaut 1" << endl;
```

le `cout` est donc situé après le `if`.

Le compilateur ne donne aucun *warning* à la compilation.

Et, si on met tout de même un point-virgule, le message "a vaut un", va être affiché, quelle que soit la valeur de a. Alors, pourquoi? Parce qu'en fait, ce point-virgule est considéré comme une instruction, une instruction qui ne fait rien. Et ce code-ci va être interprété de la façon suivante, c'est-à-dire que le point-virgule, cette instruction qui ne fait rien, va être considérée à l'intérieur du branchement conditionnel, et l'instruction cout va être considérée après le branchement conditionnel.

notes

résumé

1m 26s

Si on utilise des accolades même quand il n'y a qu'une instruction dans le bloc, et qu'on écrive le test de la façon suivante:

```
if (a == 1) {
  cout << "a vaut 1" << endl;
}
```

l'erreur précédente a beaucoup moins de chance d'arriver.

C'est-à-dire que si a vaut un, on va exécuter l'instruction vide, c'est-à-dire, ne rien faire. Et, quelqu'il, quelque soit la valeur de a, exécuter l'instruction cout, c'est-à-dire afficher le message "a vaut un". Et attention, dans ce cas, le compilateur ne donne aucun warning à la compilation. Vous remarquerez que si on avait utilisé des accolades, alors qu'il n'y a qu'une seule instruction dans le branchement conditionnel, l'erreur que nous venons de voir aurait eu beaucoup moins de chance d'arriver. Donc, c'est une bonne idée d'utiliser systématiquement des accolades, même

notes

résumé

2m 1s

Erreurs classiques

Ne pas oublier les accolades, l'indentation ne suffit pas:

```
if (n < p)
    cout << "n est plus petit que p" << endl;
    max = p;
else
    cout << "n est plus grand ou egal a p" << endl;
```

quand il n'y a qu'une seule instruction dans le bloc. Une troisième erreur possible est justement, d'oublier les accolades, comme dans ce code-ci. Le fait d'avoir indenté ces lignes, c'est-à-dire, de les avoir décalées légèrement vers la droite, ne suffit pas à ce qu'elles soient dans le branchement conditionnel, contrairement à d'autres langages. Et à votre avis, que ce passe- t-il, si j'essaye de compiler ce code?

notes

résumé

2m 37s

Erreurs classiques

Ne pas oublier les accolades, l'indentation ne suffit pas:

```
if (n < p)
  cout << "n est plus petit que p" << endl;
  max = p;
else
  cout << "n est plus grand ou egal a p" << endl;
```

génère à la compilation l'erreur:

```
syntax error before "else"
```

Et bien le compilateur va m'afficher ce message d'erreur, c'est-à-dire qu'il pense qu'il y a une erreur de syntaxe, avant le mot-clé else. Alors, pourquoi? C'est parce qu'en fait, ce code est

notes

résumé

3m 1s

Erreurs classiques

Ne pas oublier les accolades, l'indentation ne suffit pas:

```
if (n < p)
  cout << "n est plus petit que p" << endl;
  max = p;
else
  cout << "n est plus grand ou egal a p" << endl;
```

génère à la compilation l'erreur:

syntax error before "else"

Voici une meilleure présentation du code précédent:

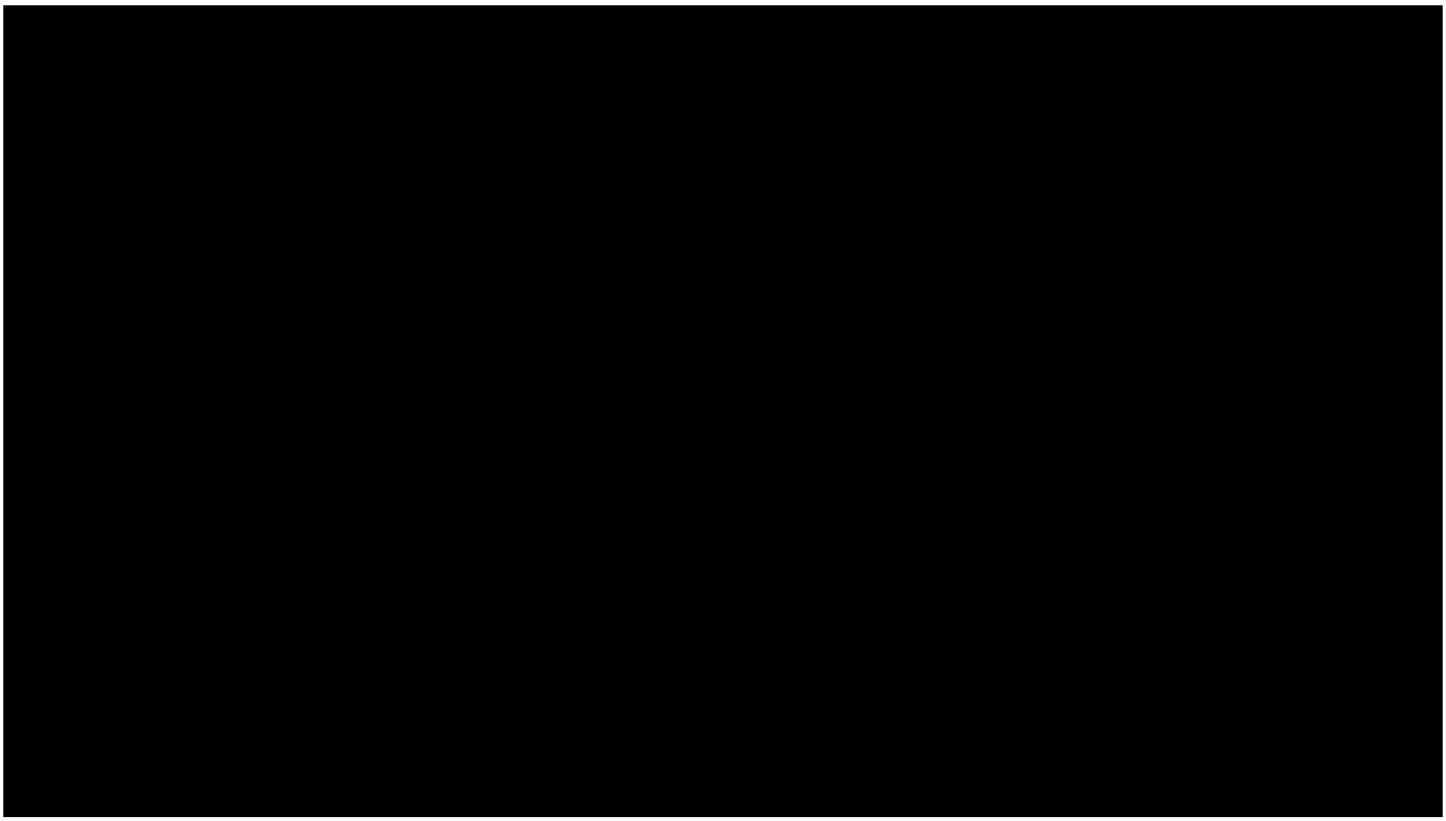
```
if (n < p)
  cout << "n est plus petit que p" << endl;

max = p;
else
  cout << "n est plus grand ou egal a p" << endl;
```

interprété de la même façon que ce code-ci. C'est-à-dire que la première instruction cout, ici, est considérée, comme à l'intérieur du branchement conditionnel. C'est ce que j'ai représenté, ici. Mais cette deuxième instruction est considérée comme après le branchement conditionnel. C'est ce que j'ai représenté, ici.

notes

résumé

3m 13s



Le compilateur tombe, ensuite, sur le mot-clé else, et pour lui, il n'y a aucune instruction if, qui se rattache à ce mot-clé else, puisqu'on est déjà sorti de ce branchement conditionnel-là. Il pense, donc, qu'il y a une erreur de syntaxe à ce niveau-ci du programme, et va afficher ce message d'erreur. Passons, maintenant, à trois quiz. Passons, maintenant, à trois quiz.

notes

résumé

3m 37s