Support de cours

Cours:

Initiation à la programmation (en C++)

Vidéo:

Erreurs de débutant, le type bool (partie 5)

Concepts (extraits des sous-titres générés automatiquement) :

Type bool. Premier branchement conditionnel. Variable de type bool. Réponse a.. Ensemble de cette condition. Type des conditions. Branchement conditionnel suivant. Variables de type int. Utilsier des opérateurs logiques. Dernière fois. Intérieur du branchement conditionnel. Valeur de cette condition. Troisième instruction. Nombreux problèmes. Exemple précédent.

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

vers la vidéo

EPFL

Erreurs de débutant
Le type `bool`
(Partie 5)

Initiation à la programmation (C++)

Vincent Lepetit, Jean-Cédric Chappelier et Jamila Sam

...

notes

résumé

0m 0s

```
cout << "Entrez le premier nombre:" << endl;
cin >> n;2
cout << "Entrez le deuxieme nombre:" << endl;
cin >> p;1

if ((n < p) and (2 * n >= p)) {
  cout << "1";
}

if ((n < p) or (2 * n >= p)) {
  cout << "2";
}

if (n < p) {
  if (2 * n >= p) {
    cout << "3";
  } else {
    cout << "4";
  }
}
cout << endl;
```

Qu'affiche ce programme quand l'utilisateur entre **2 et 1** ?

Rappel:

- Pour le ET (`and`):
les deux conditions doivent être vraies;

- Pour le OU (`or`):
au moins l'une des conditions doit être vraie.

A: `2`
B: `24` ?
C: `123`
D: `1234`

C'est la réponse A.

notes

résumé

0m 1s

```
cout << "Entrez le premier nombre:" << endl;
cin >> n;
cout << "Entrez le deuxieme nombre:" << endl;
cin >> p;

if ((n < p) and (2 * n >= p)) {
  cout << "1";
}
if ((n < p) or (2 * n >= p)) {
  cout << "2";
}
if (n < p) {
  if (2 * n >= p) {
    cout << "3";
  } else {
    cout << "4";
  }
}
cout << endl;
```

Qu'affiche ce programme quand l'utilisateur entre **2 et 1** ?

A: 2
B: 24
C: 123
D: 1234

Pour cette dernière fois, l'utilisateur rentre la valeur deux pour n, et la valeur un pour p. Cette condition est donc fausse, puisque deux n'est pas strictement inférieur à un. Comme l'ensemble de cette condition utilise un et, je n'ai pas besoin de considérer la valeur de cette condition. Je sais que le tout est forcément faux. On saute, donc, ce premier branchement conditionnel. Cette condition est toujours fausse. Par contre, cette fois-ci, j'utilise un ou. Donc, je suis obligée de considérer la valeur de cette condition. Cette expression vaut deux fois n. C'est-à-dire, deux fois deux, c'est à dire, quatre. p vaut un. Donc, cette condition est vraie, puisque quatre est supérieur ou égal à un. Donc, l'ensemble de cette condition est vraie. On va, donc, exécuter cette instruction, et afficher deux. On passe, ensuite, au branchement conditionnel suivant. Cette condition, elle est toujours fausse. On va, donc, sauter ce qu'il y a, à l'intérieur du branchement conditionnel,

notes

résumé

0m 5s

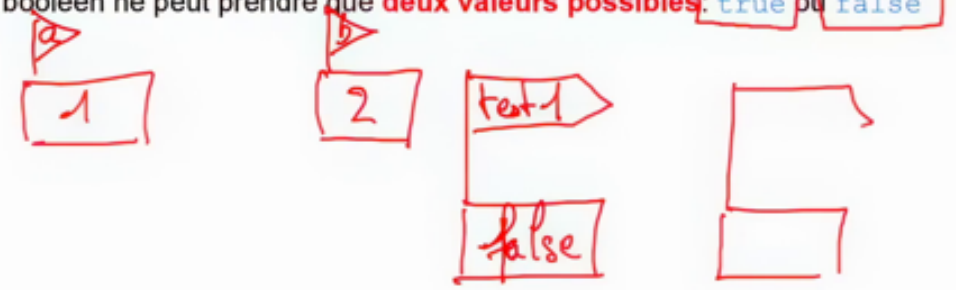
Le type bool

Le type `bool` (pour *boolean*, ou booléen) est le type des **conditions**.

Il permet de déclarer des **variables contenant la valeur d'une condition**.

Une variable de type booléen est souvent appelée simplement un **booléen**.

Comme les conditions, un booléen ne peut prendre que **deux valeurs possibles**: `true` ou `false`



Exemple:

```
int a(1);
int b(2);
bool test1(a == b);
bool test2(a < b);
```

Faux

passer à ce stade du programme, qui fait simplement un retour à la ligne. Terminons avec le type bool, pour booléen, qui est le type des conditions. Ce type va donc nous permettre de déclarer des variables qui vont contenir la valeur d'une condition. Une variable de type bool est souvent appelée un booléen. Et un booléen ne peut donc prendre que deux valeurs, soit vrai, soit faux, ou plus exactement, en C++, on va utiliser les valeurs littérales, true et false. Dans cet exemple, je commence par déclarer deux variables de type int. a qui sera initialisée à un. b qui est initialisée à deux. Dans cette troisième instruction, je déclare une variable qui s'appelle test un, qui est de type bool. Comme toutes les variables, je peux représenter test un avec une boîte. Et test un sera initialisé à la valeur de cette condition: a égal b. Comme a vaut un, b vaut deux, cette condition est fausse. Et test un sera donc initialisé à faux, ou plus exactement false, qui veut dire faux, en anglais. Cette dernière instruction déclare une variable test2,

notes

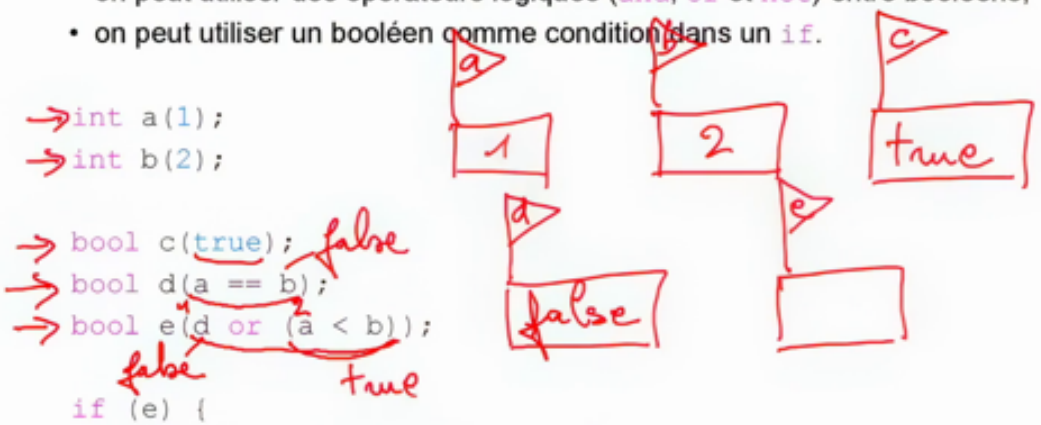
résumé

1m 25s

On peut initialiser des booléens à l'aide des constantes `false` et `true`.

On peut utiliser des booléens comme des conditions. Par exemple:

- on peut utiliser des opérateurs logiques (`and`, `or` et `not`) entre booléens;
- on peut utiliser un booléen comme condition dans un `if`.




```
int a(1);
int b(2);

bool c(true);
bool d(a == b);
bool e(d or (a < b));

if (e) {
  cout << "e vaut true" << endl;
}
```

de type bool. Un booléen, donc. Et, test2 va être initialisé à la valeur de cette condition. a vaut un, b vaut deux. Cette condition est donc vraie. test2 va, donc, être initialisé à vrai, ou plus exactement à true, qui veut dire vrai, en anglais. Certains étudiants, donc, ont un problème avec les booléens, d'autres font parfois un bloquage. Alors, rappellez-vous qu'une variable de type bool, est une variable comme les autres. C'est-à-dire que vous pouvez vous la représenter comme une boîte. Il se trouve juste que cette boîte ne peut contenir que deux valeurs possibles, soit true, soit false. On peut également, utilsier des opérateurs logiques, et ou non, entre variables de type bool, et on peut utiliser aussi ces variables, comme des conditions dans un branchement conditionnel, et c'est ce que nous allons voir, dans cet exemple. Comme dans l'exemple précédent, je commence par déclarer une variable appelée a, de type int, et initialisée à la valeur un, et une variable b, de type int, également, initialisée à la valeur deux. Dans cette instruction-ci, je déclare une variable de type bool, qui s'appelle c, et initialisée à la valeur true. Ensuite, ici, je déclare une valeur de type bool qui s'appelle d, initialisée à la valeur de la condition, a égal b. a vaut 1, b vaut 2. La condition est donc fausse, et vaut false. d est donc initialisée à false. Ensuite, ici, je déclare une variable de type bool, appelée e, initialisée à la valeur de cette condition. d vaut false. Cette condition, a inférieure à b, est vraie.

notes

résumé

3m 1s

Elle vaut donc true. L'opérateur logique est, ici, un or. Donc, toute cette condition est vraie, et vaut donc, true. e est donc initialisée à true. Ensuite, dans ce branchement conditionnel, la condition est la valeur de la variable e de type bool. e vaut true. La condition est donc vraie, et on va rentrer dans le branchement conditionnel, pour exécuter cette instruction, qui va afficher le message, e vaut true. C'est tout pour le moment, pour les booléens. Mais, les booléens sont utiles pour de nombreux problèmes, et nous les rencontrerons dans des exemples concrets, dans la suite du cours. concrets, dans la suite du cours.

notes

résumé

5m 25s