Support de cours

Cours:

Initiation à la programmation (en C++)

Vidéo:

Définitions (partie 3)

Concepts (extraits des sous-titres générés automatiquement) :

Exemple de la fonction. Initialisation de la variable n. Instruction return. Cas particulier. Clavier de la valeur. Fait possible. Valeur négative. Type de retour de la fonction. Dernière remarque. Expression return. Compilateur. Fonction. Bonne condition. Test. Nombre.

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

vers la vidéo

EPFL

Fonctions : définitions
(Partie 3)

Initiation à la programmation (C++)

Vincent Lepetit, Jean-Cédric Chappelier et Jamila Sam

...

notes

résumé

0m 0s

Remarques sur l'instruction `return` (4/4)

EPFL

Le compilateur doit être sûr de toujours pouvoir exécuter un `return`:

```
double lire() {
  cout << "Entrez un nombre : ";
  double n(0.0);
  cin >> n;
  if (n > 0.0) {
    return n;
  }
  // Erreur : pas de return si n <= 0 !
}
```

n < 0 ?

Quatrième et dernière remarque concernant l'instruction return. Le compilateur doit toujours pouvoir exécuter un return. Je reprends ici l'exemple de la fonction lire écrite un tout petit peu différemment. On retrouve ici la demande d'entrer un nombre à l'utilisateur, la déclaration et initialisation de la variable n, la lecture sur le clavier de la valeur de n mais, ici, on a donc simplement un test si n est strictement positif alors, la fonction va exécuter ce return et va donc terminer et renvoyer la valeur de n. Mais, que se passe t-il si n est négatif ? C'est tout à fait possible que n soit négatif, on pourrait tout à fait imaginer qu' à ce niveau-là, l'utilisateur est rentré une valeur négative. A ce compte-là, le test ici va être faux et donc la fonction va sauter après le if et va se retrouver ici, et vous voyez qu'à partir de là, il n'y a plus rien à exécuter. On va donc terminer la fonction sans return ceci n'est pas possible et donc le compilateur va vous renvoyer une erreur. Le compilateur vérifie que l'on doit terminer la fonction chaque fois en ayant une expression return

notes

résumé

0m 1s

Remarques sur l'instruction `return` (4/4)

Le compilateur doit être sûr de toujours pouvoir exécuter un `return`:

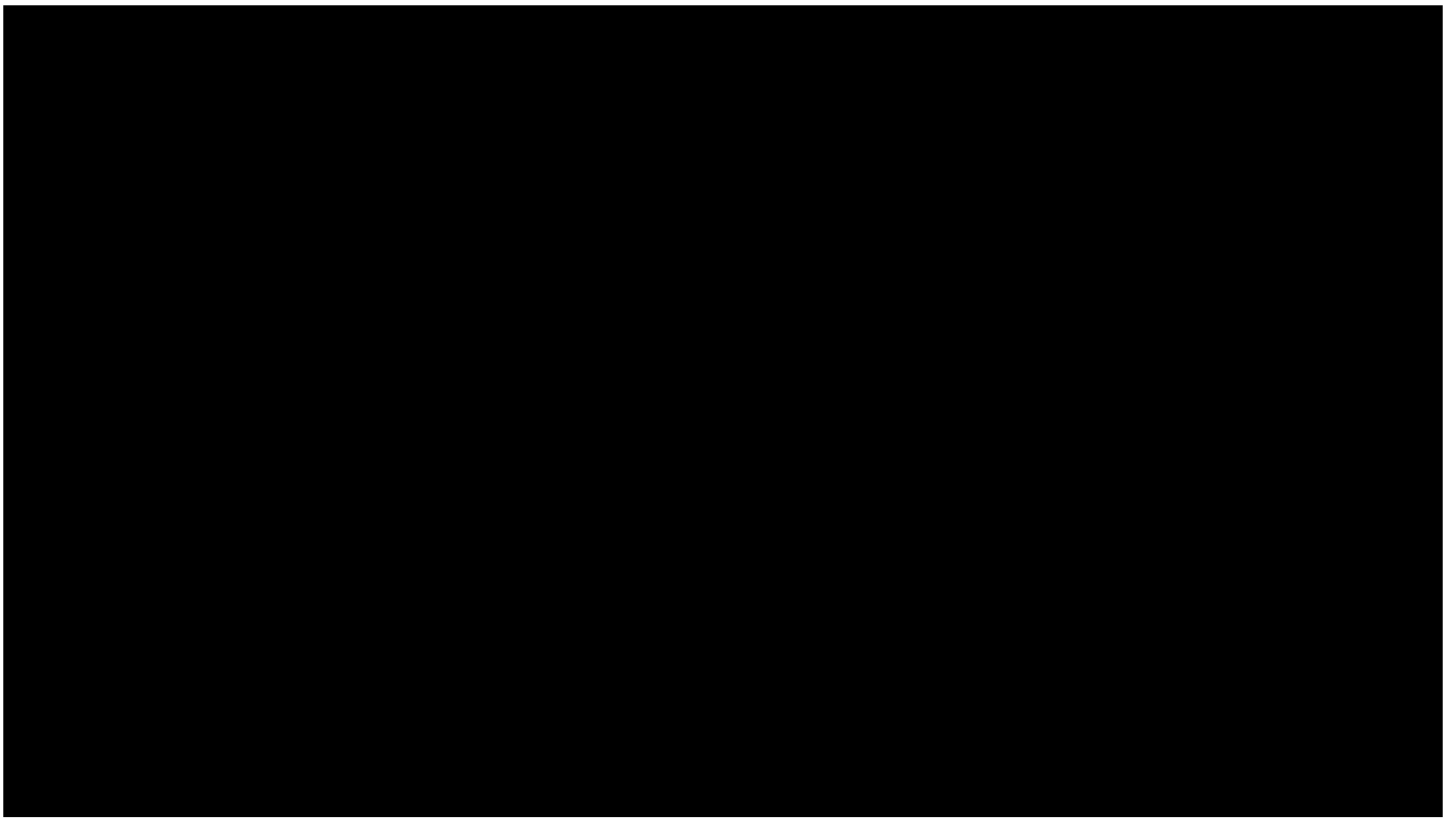
```
double lire() {
  cout << "Entrez un nombre : ";
  double n(0.0);
  cin >> n;
  if (n > 0.0) {
    return n;
  }
  // Erreur : pas de return si n <= 0 !
}
```

qui retourne un double puisque c'est le type de retour de la fonction. Dans ce cas particulier, ce que l'on aurait pu faire c'est corriger par exemple la fonction et éventuellement rajouter ici une boucle

notes

résumé

1m 25s



qui reprend tout ce bloc ici, tant que par exemple, la valeur de n est négative ou nulle, on répète alors le bloc qui demande la question et qui fait saisir un autre à un nombre ce qui aurait permis de tout le temps terminer avec une bonne condition. terminer avec une bonne condition.

notes

résumé

1m 37s