Support de cours

Cours:

Initiation à la programmation (en C++)

Vidéo:

Boucles conditionnelles (partie 1)

Concepts (extraits des sous-titres générés automatiquement) :

Nombre de conditions. Boucles conditionnelles. Nombre de notes. Instructions particulières. Forme de boucles do. Nombre de traitements. Types d'instructions de répétition. Boucle do while. Nombre de notes voulu. Notion d'itération. Évaluation de cette condition. Boucle for. Nombre de répétitions. Observation importante. Éléments de la syntaxe d'une instruction.

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

vers la vidéo

EPFL

Boucles conditionnelles
(Partie 1)

Initiation à la programmation (C++)

Vincent Lepetit, Jean-Cédric Chappelier et Jamila Sam

...

notes

résumé

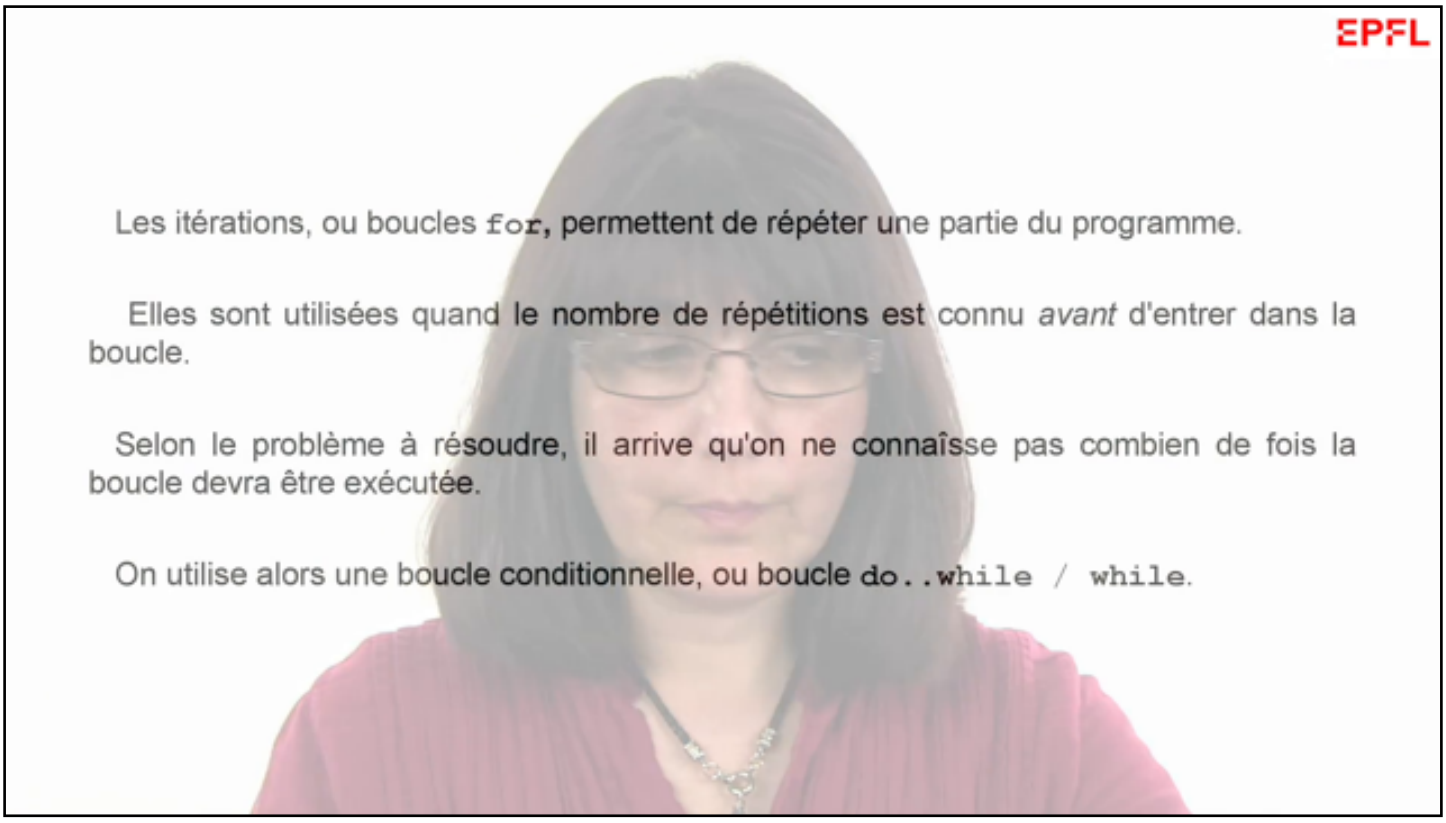
0m 0s

Dans des séquences précédentes, nous avons

notes

résumé

0m 1s



abordé la notion de structures de contrôle. Nous avons vu qu'il s'agit d'instructions particulières qui vont permettre de répéter des traitements, ou d'exécuter des traitements en fonction d'un certain nombre de conditions. Nous allons aujourd'hui continuer sur cette lancée et aborder une autre façon de répéter des traitements que l'on appelle des boucles conditionnelles.

notes

résumé

0m 5s

Les itérations, ou boucles `for`, permettent de répéter une partie du programme.

Elles sont utilisées quand le nombre de répétitions est connu *avant* d'entrer dans la boucle.

Selon le problème à résoudre, il arrive qu'on ne connaîsse pas combien de fois la boucle devra être exécutée.

On utilise alors une boucle conditionnelle, ou boucle `do..while` / `while`.

Lorsque l'on souhaite, dans un programme, répéter un certain nombre de traitements et que le nombre de répétitions est connu à priori, c'est-à-dire avant même d'entrer dans la boucle, nous avons vu qu'il était possible d'avoir recours à une boucle for, la notion d'itération. Il existe cependant des situations où l'on souhaite répéter un traitement. Le répéter, par exemple tant qu'une condition est vérifiée mais on ne sait pas combien de fois il va être nécessaire de répéter les traitements jusqu'à ce que la condition ne soit plus vérifiée et dans ce cas, il faut avoir recours à d'autres types d'instructions de répétition: les boucles conditionnelles, les boucles dont l'exécution dépend de la réalisation d'une condition et plus concrètement en

notes

résumé

0m 24s

```
do {
  cout << "Entrez le nombre de notes:" << endl;
  cin >> nombre_de_notes;
} while(nombre_de_notes <= 0);
```

C++, elles prendront la forme de boucles do while ou while. Pour illustrer notre propos, reprenons notre petit exemple de calcul des moyennes d'un certain nombre de notes. Donc nous ne savons pas à priori de combien de notes nous voulons calculer la moyenne, donc nous commençons par le demander à l'utilisateur et le nombre de notes voulu va être saisi via une interaction au clavier. Ensuite, si ce nombre de notes est positif, nous allons utiliser une itération for pour saisir toutes les notes. Donc à chaque itération demandée, la note numéro i, saisir cette note via une interaction au clavier et incorporer la note à la somme de toutes les notes. Donc une fois que nous avons atteint le nombre de notes voulues, nous sortons évidemment de l'itération et nous continuons l'exécution du programme en séquence et à ce moment-là, nous allons calculer la moyenne des notes et la faire afficher. Une observation que l'on peut faire ici est qu'il n'est pas très naturel qu'un utilisateur saisisse un nombre de notes qui soit nul et la question que l'on peut se poser, c'est dans ce cas-là, comment forcer l'utilisateur à introduire systématiquement un nombre de notes qui soit supérieur à zéro?. Très concrètement, nous voulons ici répéter ces instructions jusqu'à ce que l'utilisateur consente à introduire un nombre de notes qui soit strictement positif. Nous ne savons pas ici combien de fois il faudra répéter ces instructions puisque nous ne pouvons pas prédire à priori quand l'utilisateur va effectivement saisir un nombre de notes strictement positif. Et donc dans ce cas, il faut avoir recours à une structure de contrôle particulière et la réponse à nos besoins ici serait ce qu'on appelle la

notes

résumé

1m 1s

```
do {
   cout << "Entrez le nombre de notes:" << endl;
   cin >> nombre_de_notes;
} while(nombre_de_notes <= 0);
```

tant que

Vraie

boucle do while La boucle do while se rédige concrètement comme suit: on utilise le mot réservé "do" suivi d'une paire d'accolades ouvrantes et fermantes et le mot réservé "while" pour dire tant que, ce qui veut dire concrètement répéter tant que cette condition est vraie.

notes

résumé

2m 37s

Pas-à-pas

```
do {
    cout << "Entrez le nombre de notes:" << endl;
    cin >> nombre_de_notes;
} while(nombre_de_notes <= 0);
```

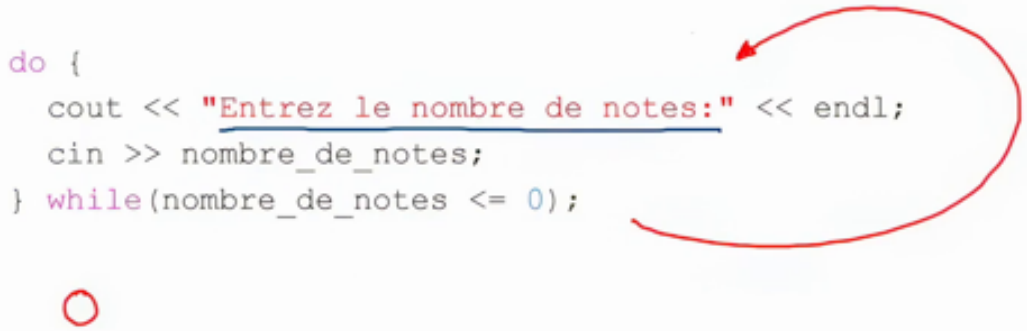
Donc très concrètement, nous allons répéter les instructions qui se trouvent dans ce que l'on appelle le corps, donc toutes les instructions comprises entre les accolades. Nous allons répéter l'exécution de ce corps tant que la condition est vraie. Examinons maintenant pas à pas le déroulement de notre petit exemple avec la boucle do while. Nous pré-supposons ici qu'une variable nombre_de_notes a été au préalable déclarée et est prête à contenir un nombre de note saisies par un utilisateur. Lorsque nous abordons cette instruction, rien ne nous empêche d'entrer dans le corps de la boucle et à ce moment-là, nous allons signifier à l'utilisateur que nous attendons de lui qu'il introduise un nombre de notes. Le nombre de notes est saisi via une interaction au clavier et nous aboutissons à l'exécution de cette ligne qui va évaluer la condition de continuation de la boucle. Supposons dans un premier temps que l'utilisateur saisisse en guise de note la valeur zéro. Donc l'évaluation de cette condition, dans ce cas-là, va retourner "true" ce qui signifie que le nombre de notes ne

notes

résumé

3m 1s

Pas-à-pas



```
do {
  cout << "Entrez le nombre de notes:" << endl;
  cin >> nombre_de_notes;
} while(nombre_de_notes <= 0);
```

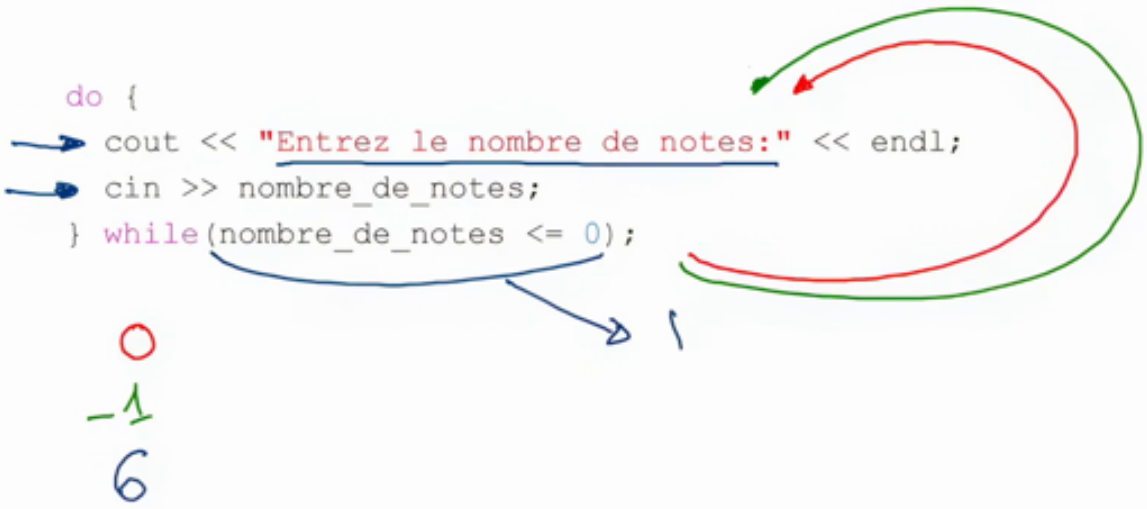
répond pas aux attentes que nous avons et nous allons donc entamer un nouveau cycle puisque la condition est vraie. Nous allons donc à nouveau exécuter le corps

notes

résumé

4m 1s

Pas-à-pas



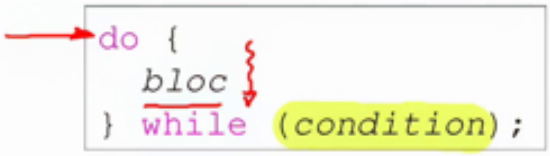
de la boucle et supposons que cette fois-ci, l'utilisateur introduise en guise de nombre de notes au moment de la saisie clavier la valeur -1. Au moment de l'évaluation de la condition de continuation, cette condition va, comme précédemment, être évaluée à "true" puisque -1 est bien évidemment inférieur ou égal à zéro, et à ce moment-là, nous allons à nouveau entamer une nouvelle itération de la boucle. Nous entamons alors une troisième itération de la boucle, à nouveau nous allons demander à l'utilisateur d'entrer un nombre de notes, lire cette note via une interaction clavier et supposons que cette fois-ci, de guerre lasse, l'utilisateur finisse par comprendre ce qu'on attend de lui et saisisse un nombre de notes qui soit strictement positif, par exemple 6. À ce moment-là, au moment de l'évaluation de la condition de continuation de la

notes

résumé

4m 10s

Syntaxe de l'instruction `do...while`



- Comme pour l'instruction `if`:
  - La condition peut utiliser des opérateurs logiques.
  - Les parenthèses autour de la condition sont obligatoires.
- Les instructions à l'intérieur de la boucle `do...while` sont toujours exécutées au moins une fois.
- Si la condition ne devient jamais fausse, les instructions dans la boucle sont répétées indéfiniment !

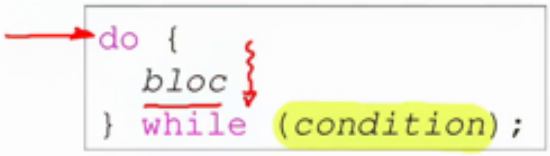
boucle, cette évaluation va retourner, cette fois fausse puis que 6 est supérieur strictement à zéro et à ce moment-là, nous allons donc sortir de la boucle do while et continuer l'exécution après le point-virgule terminant l'instruction do while. Donc voici maintenant formellement les éléments de la syntaxe d'une instruction do while en C++. Les mots réservés do while qui encadrent ce que l'on appelle le corps de do while, lequel est entre accolades ouvrantes et fermantes. Après le while se trouve la condition de continuation de la boucle et la sémantique est la suivante: les traitements du corps sont répétés tant que la condition est évaluée à "true". Donc on répète les traitements tant que la condition est évaluée à "true". Donc comme pour l'instruction de branchement conditionel if, la condition d'arrêt de la boucle do while peut être rédigée de façon relativement complexe et avoir recours à des opérateurs logiques. On peut, par exemple, écrire des conditions qui auraient cette allure, donc on itère les traitements, on répète les traitements tant que x est égal à y plus z et que z est supérieur à 10 ou que t est inférieur à 25, par exemple. Donc on peut imaginer la rédaction de conditions d'arrêt relativement sophistiquées notamment en ayant recours à des connecteurs logiques, à des opérateurs logiques. Il faut noter également que les parenthèses autour de la condition sont des éléments de syntaxe obligatoires en C++ et donc tout comme pour le if, il va falloir rédiger notre condition entre une paire de parenthèses. Une observation importante à faire à propos de la boucle do while est que son corps va systématiquement être exécuté au moins une fois. Effectivement, si on se rappelle l'exemple pas à pas d'exécution d'une boucle do while, arrivé à ce stade de l'exécution rien ne nous empêche de rentrer à l'intérieur du corps et donc d'exécuter ce bloc au moins une fois.

notes

résumé

5m 1s

Syntaxe de l'instruction `do...while`



- Comme pour l'instruction `if`:
  - La condition peut utiliser des opérateurs logiques.
  - Les parenthèses autour de la condition sont obligatoires.

- Les instructions à l'intérieur de la boucle `do...while` sont toujours exécutées au moins une fois.

- Si la condition ne devient jamais fausse, les instructions dans la boucle sont répétées indéfiniment !

Ce bloc d'instructions ne sera exécuté qu'une seule fois si au bout de la première itération, l'évaluation de cette condition est

notes

résumé

telle que le résultat de l'évaluation est fausse. À ce moment-là, nous avons vu que nous allons poursuivre l'exécution après la boucle. Mais entre-temps, le bloc d'instructions ici aura été exécuté au moins une fois. Donc ceci est important et fait partie des caractéristiques de la boucle do while. Autre observation qui a son importance: si la condition ne devient jamais fausse, et bien les traitements sont répétés indéfiniment. Donc il faut être attentif à bien formuler la condition d'arrêt de sorte à ne pas tomber dans une boucle infinie qui généralement n'est pas un comportement que l'on va souhaiter, en tout cas à ce stade de votre apprentissage. Voilà pour ce qui est des éléments fondamentaux concernant la boucle do while. Avant de clore sur le sujet de la syntaxe de l'instruction do while, une dernière toute petite mise en garde syntaxique: le petit point-virgule qu'on a tendance à oublier et qui clôt l'instruction do while. l'instruction do while.

notes

résumé

7m 1s