Support de cours

Cours:

Initiation à la programmation (en C++)

Vidéo:

Méthodologie

Concepts (extraits des sous-titres générés automatiquement) :

Arguments de type double. Type de retour. Types de passage. Nombres réels. Quatrième étape. Fonction. Séquence vidéo. Exemple de notre fonction moyenne. Racine carrée d'un double. Façon optionnelle. Fonction sqrt. Bonne façon. Valeur de la racine carrée. Telle expression. Corps de la fonction.

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

vers la vidéo

EPFL

Fonctions : méthodologie
(Partie 1)

Initiation à la programmation (C++)

Vincent Lepetit, Jean-Cédric Chappelier et Jamila Sam

...

notes

résumé

0m 0s

Pour résumer : Méthodologie pour construire une fonction

EPFL

1. clairement identifier ce que **doit faire** la fonction
   (ce point n'est en fait que conceptuel, on n'écrit aucun code ici !)
   - ☞ ne pas se préoccuper ici du ~~*comment*~~, mais bel et bien du **quoi** !

     Les instructions dans le corps de la fonction dont la finalité n'est pas le calcul de la valeur de retour, ou qui modifient des objets extérieurs à la fonction (non passés en paramètre), sont appelées des **« effets de bord »**.

sqrt(x)

Je voudrais dans cette séquence vidéo, résumer la méthodologie générale que je vous conseille de suivre pour concevoir, écrire, une fonction. Donc tout d'abord, je vous demande de clairement identifier ce que la fonction doit faire. C'est vraiment le point important. Il faut commencer par identifier ce que la fonction doit faire. Ça ne sert à rien d'essayer d'écrire une fonction si vous savez pas clairement ce que vous voulez faire. Donc c'est vraiment ici un point conceptuel. Il n'y a aucune ligne de c++ à écrire ici. Mais il y a vraiment à concevoir, éventuellement en vous aidant d'un papier et d'un crayon en faisant des schémas, à concevoir ce que la fonction doit faire. On s'intéresse ici vraiment à ce que fait la fonction. On s'intéresse au "quoi" et on s'intéresse absolument pas pour l'instant à "comment" faire ce que la fonction doit faire. Et c'est là où il faut bien faire attention de ne pas faire ce que l'on a appelé précédemment dans la vidéo sur les prototypes "des effets de bord". C'est à dire que la fonction doit faire exactement ce pourquoi elle a été prévue, ni plus ni moins. Si par exemple vous avez conçu une fonction sqrt dont le but est de calculer la racine carrée d'un double par exemple ici x

notes

résumé

0m 1s

Pour résumer : Méthodologie pour construire une fonction

1. clairement identifier ce que **doit faire** la fonction
2. quels **arguments** ?
   ☞ que doit recevoir la fonction pour faire ce qu'elle doit ?

et dont l'appel correspondrait par exemple comme ça à z=sqrt(x) La fonction ici sqrt doit simplement calculer la racine carrée de x et la retourner à z. Elle ne doit pas polluer l'affichage sur le terminal avec des valeurs affichées. Elle doit juste calculer, sans afficher, la valeur de la racine carrée de x.

notes

résumé

1m 13s

Pour résumer : Méthodologie pour construire une fonction

1. clairement identifier ce que **doit faire** la fonction
2. quels **arguments** ?
3. passage(s) par valeur(s) / référence(s) ?

   ☞ pour chaque argument : doit-il être modifié par la fonction ?
   (si oui : passage par référence)

moyenne (

Ensuite, quand vous avez fait ceci, vous devez vous demander : qu'est-ce que la fonction doit recevoir ? Quels sont les arguments que la fonction doit recevoir ? Si par exemple je reprends l'exemple de notre fonction moyenne. Donc très clairement au début, j'ai pensé que c'était une fonction qui devait calculer la moyenne de 2 nombres réels. Elle prendra 2 nombres réels et elle calculera la moyenne de 2 nombres réels. Donc ici, à ce stade-là, je me dis qu'elle va recevoir deux arguments de type double. Par exemple je peux les appeler x et y. Ensuite vous allez vous demander si vous devez passer vos arguments par valeur ou par référence. Je vous rappelle brièvement qu'il existe deux types de passage : le passage par valeur, qui fait une copie et ne permet donc pas à la fonction de modifier les arguments reçus et le passage par référence qui sera utilisé lorsque on veut modifier les arguments reçus passés à la fonction. Donc si je prends l'exemple ici d'un appel à la fonction moyenne,

notes

résumé

1m 36s

Pour résumer : Méthodologie pour construire une fonction

1. clairement identifier ce que **doit faire** la fonction — QUOI ?
2. quels **arguments** ?
3. passage(s) par valeur(s) / référence(s) ?
   Optionnel : se demander si cela a un sens de donner une valeur par défaut au paramètre correspondant
4. quel type de retour ?
5. (maintenant, et seulement maintenant) Se préoccuper du *comment* :
   comment faire ce que doit faire la fonction ?
   ☞ c'est-à-dire écrire le corps de la fonction

```
double moyenne(double x, double y)
{
    double resultat(0.0);
    resultat = (
    return resultat;
}
```

où on passerait ici deux arguments a et b. La question : est-ce que moyenne va modifier a ou modifier b ? La réponse ici est clairement : non. Et donc ici, on va faire un passage par valeur. Si par contre vous pensez que votre fonction doit modifier les arguments qu'elle a reçu, alors à ce moment-là il faudra faire un passage par référence comme on a vu dans la vidéo concernant l'appel. Ensuite vous pouvez de façon optionnelle vous demander si ça a un sens de donner une valeur par défaut aux paramètres correspondants. Ceci sera traité dans une autre vidéo plus tard, mais c'est à cet endroit-là que vous vous poseriez la question. Quatrième étape : vous demander de quel type doit-être la valeur que la fonction doit retourner au reste du programme. Donc pour ça, vous écrivez le code comme ça dans votre tête z = "appel de votre fonction avec les différents arguments" Est-ce que ça a un sens d'écrire z = "un appel de la fonction" ? Donc par exemple, est-ce que ça a un sens d'écrire z = "la moyenne de a et b" ? Dans notre cas évidemment ça a un sens, et le sens c'est que z à ce moment-là sera la moyenne de a et b. Donc si ça a un sens, le type de retour doit être le type de z pour lequel cette expression a un sens. Donc ici ça sera typiquement un double et donc ça veut dire que la fonction moyenne doit retourner un double. Si une telle expression n'a pas de sens, si ça n'a absolument pas de sens d'écrire z = "appel de fonction", par exemple est-ce que ça aurait un sens d'écrire z = "affiche", on avait vu tout à l'heure "afficher racine". Est-ce que ça a un sens d'écrire z = "affiche racine de 2" ? Attention ici c'était bien "afficher". C'était

notes

résumé

2m 37s

Pour résumer : Méthodologie pour construire une fonction

1. clairement identifier ce que **doit faire** la fonction
2. quels **arguments** ?
3. passage(s) par valeur(s) / référence(s) ?
   Optionnel : se demander si cela a un sens de donner une valeur par défaut au paramètre correspondant
4. quel type de retour ?
5. (maintenant, et seulement maintenant) Se préoccuper du *comment* :
   comment faire ce que doit faire la fonction ?
   ☞ c'est-à-dire écrire le corps de la fonction

QUOI ?

```
double moyenne(double x, double y)
{
    double resultat(0.0);
    resultat = (
    return resultat;
}
```

pas "calculer la racine de 2". Qu'est-ce que serait z dans ce cas-là ? Moi je vois pas très bien à quoi ça pourrait faire référence. "Afficher" affiche et puis c'est tout, n'a rien à retourner ici. Donc cette expression à mon avis non, n'aurait pas de sens et dans ce cas-là le type de retour est à ce moment-là, on appelle ça une procédure, donc le type de retour est à ce moment-là void. Cinquième et dernière étape enfin. Maintenant et c'est seulement maintenant, pas avant, vous vous préoccupez du "comment" : comment écrire le corps de la fonction, comment la fonction va faire ce qu'elle a à faire. Au préalable on s'était uniquement préoccupé du "quoi" et c'est seulement ici, tout à la fin, qu'on se préoccupe du "comment". Si je vous donne l'exemple de la fonction moyenne dont je sais, par les étapes précédentes, qu'elle doit retourner un double. Elle s'écrit moyenne, elle reçoit un argument x par valeur, puisque cet argument doit pas être modifié, elle reçoit un deuxième argument y. On va s'intéresser maintenant à comment écrire cette fonction. Donc ici je vous conseille de commencer par, par exemple, déclarer une variable que l'on va appeler résultat, que l'on va initialiser à une valeur qui fait sens, disons ici 0, et tout de suite écrire un return de 7 qui utilise cette variable. Je vous encourage à procéder de la sorte à chaque fois que vous avez comme ça, une fonction qui retourne un type définit, donc par exemple ici un double, à définir ici tout de suite une variable que vous allez utiliser comme valeur de retour. Comme ça, ça vous facilite l'écriture de ce corps. Et ensuite, donc maintenant, on va se demander comment on calcule ce résultat. Donc ici on pourrait écrire

notes

résumé

résultat = (x + y) / 2 Supposons que l'on veuille faire une fonction

notes

résumé

6m 25s

demander_nb.cc

```
1  /* Fonction qui demande un nb compris entre 2 bornes.
2   * Exemple : un entier compris entre 1 et 10. */
3
4  demander_nb()
```

qui demande un nombre à l'utilisateur compris entre deux bornes. Première étape, c'est donc de spécifier clairement ce que doit faire la fonction. Pour cela, on peut par exemple commencer par faire un commentaire qui va indiquer ce que fait la fonction. Donc ici on va indiquer que c'est une fonction qui demande un nombre compris entre deux bornes, par exemple un entier compris entre 1 et 10. Donc on va nommer cette fonction, par exemple demander nombre On suppose qu'on est donc maintenant

notes

résumé

6m 32s

demander_nb.cc

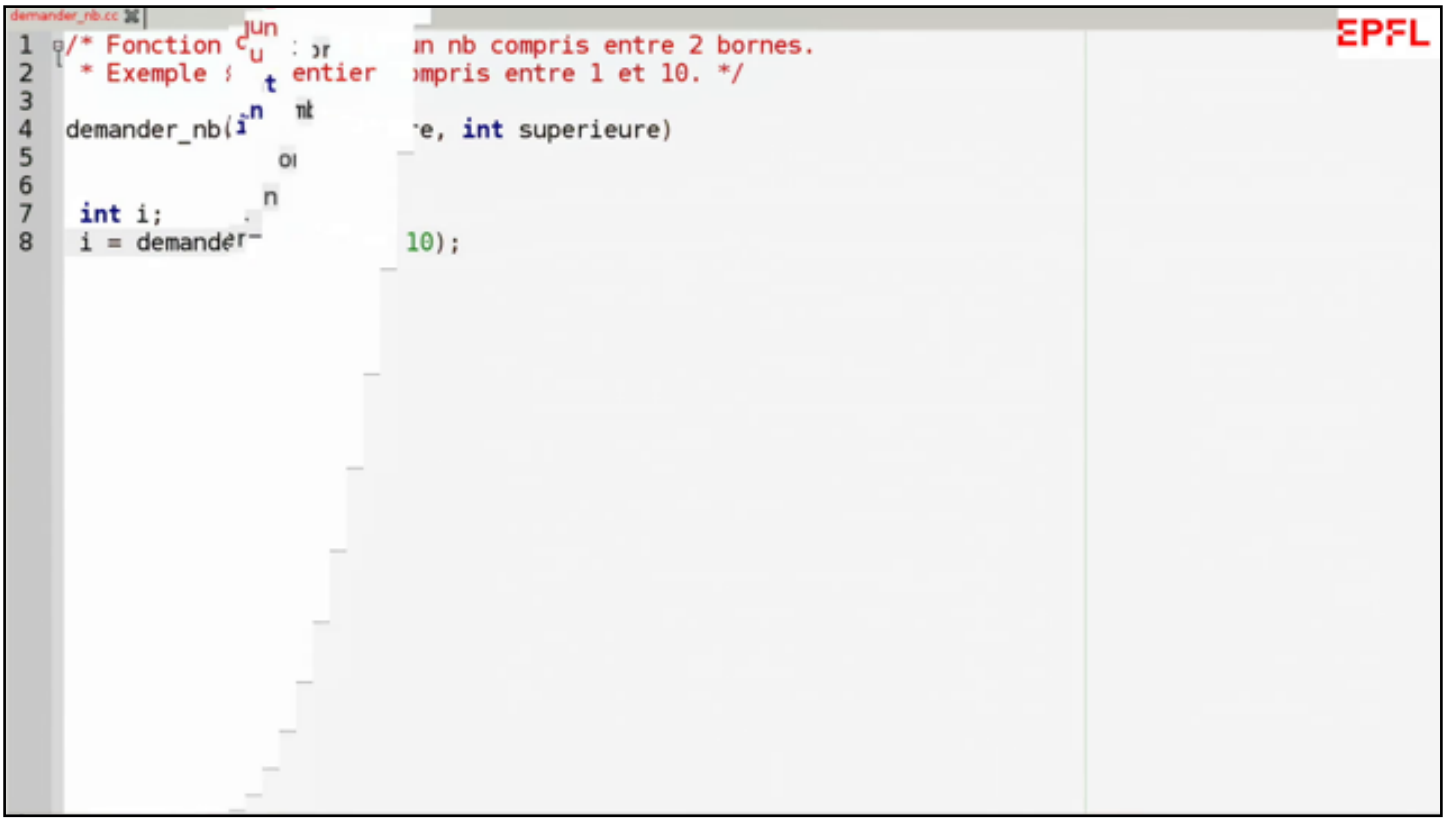
```
1 /* Fonction qui demande un nb compris entre 2 bornes.
2  * Exemple : un entier compris entre 1 et 10. */
3
4 demander_nb(int inferieure, int superieure)
```

assez au clair avec ce que doit faire la fonction et on va donc pouvoir passer à la deuxième étape, qui est d'indiquer les paramètres de la fonction. Donc ici comme on veut demander un nombre entre deux bornes, les paramètres que doit recevoir la fonction pour travailler, ce sont bien sûr les deux bornes. Donc supposons qu'on travaille sur des entiers ici. Donc les deux bornes seront un entier "borne inférieure" et puis un entier "borne supérieure". Donc on doit recevoir ici deux entiers.

notes

résumé

7m 13s



Ensuite, il faut se demander si les 2 paramètres que l'on a passé doivent être modifiés ou non par la fonction. Si on doit les passer par valeur ou par référence. Il semble clair ici que la borne inférieure et la borne supérieure ne seront pas modifiées par le fait de demander un nombre, et donc on va les passer comme indiqué ici les passer par valeurs. On peut alors passer maintenant à l'étape numéro 4, qui est de savoir quel doit-être le type de retour de la fonction. Une bonne façon de répondre à cette question, c'est de se poser la question : comment serait appelée la fonction ? Et là typiquement, la fonction serait appelée avec un entier i et on ferait un appel comme ça : i = demander_nombre (pris entre, on avait dit par exemple 1 et 10)

notes

résumé

7m 49s

demander_nb.cc

```
1  /* Fonction qui demande un nb compris entre 2 bornes.
2   * Exemple : un entier compris entre 1 et 10. */
3
4  demander_nb(int inferieure, int superieure)
5
6
7   int i;
8   i = demander_nombre(1, 10);
```

Lorsque l'on a comme ça des affectations qui peuvent récupérer la valeur de retour, où ça a un sens de récupérer la valeur de retour d'une fonction, alors naturellement, le type de retour de la fonction doit être le type qu'aurait une variable dans une telle affectation.

notes

résumé

8m 43s

Donc ici, ça aurait un sens d'écrire ce que j'ai écrit. Le type de i serait un int, ce qui fait que le type de retour de la fonction que nous allons indiquer va être un type entier. On peut donc revenir ici dans notre programme, et indiquer le type de retour, comme étant un entier. Je peux donc maintenant passer à la dernière étape. Maintenant et seulement maintenant, je me préoccupe d'écrire le corps de la fonction. Donc pour ceci je commence par écrire le corps et, un petit truc ici, lorsque vous avez une valeur de retour à renvoyer par la fonction, je vous conseille de directement déclarer une variable qui va être utilisée pour le retour. Le mieux c'est aussi de l'initialiser directement et de tout de suite indiquer vous allez retourner cette valeur. Et vous pouvez donc maintenant vous concentrer sur ici écrire exactement le corps de la fonction, vous préoccuper maintenant du "comment". Comment faire pour demander un nombre compris entre la borne inférieure et la borne supérieure. Je vous laisse le faire en guise d'exercice. en guise d'exercice.

notes

résumé

9m 1s