Support de cours

Cours:

Initiation à la programmation (en C++)

Vidéo:

Prototypes (partie 2)

Concepts (extraits des sous-titres générés automatiquement) :

Argument de type. Fonction racine carrée. Type de retour. Nom pertinent. Appel de la fonction. Début de la ligne. Dernier conseil. Double x. Racine d'un nombre. Point virgule. Prototype de votre fonction. Valeur de retour. Exemple précédent. Expression de type. Appel de fonction.

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

vers la vidéo

EPFL

Fonctions : prototypes
(Partie 2)

Initiation à la programmation (C++)

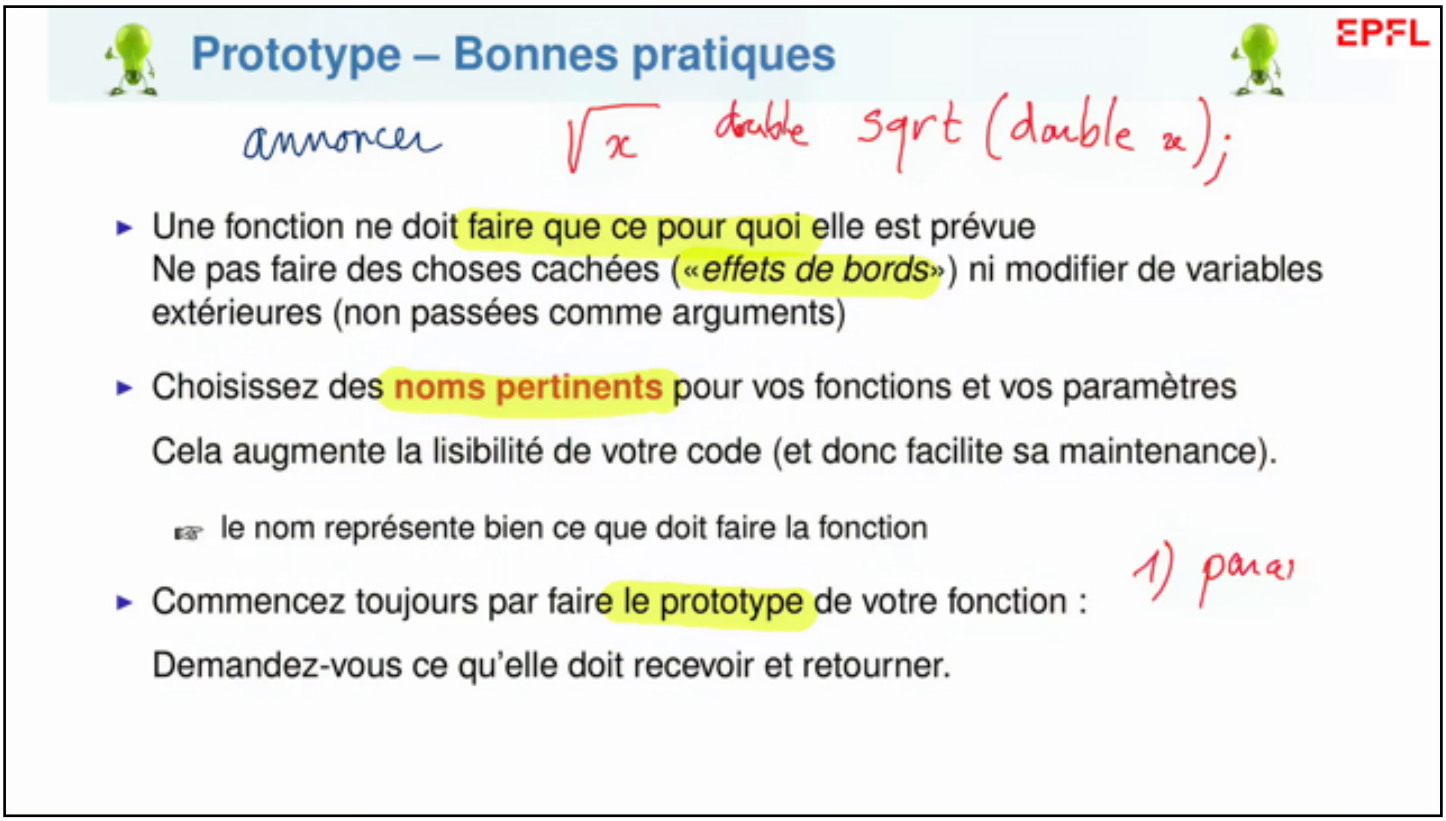
Vincent Lepetit, Jean-Cédric Chappelier et Jamila Sam

...

notes

résumé

0m 0s



Le prototype sert donc à annoncer au reste du programme ce que va faire la fonction. Pour cela, il faut donc que la fonction ne fasse que ce pour quoi elle a été prévue. Il ne faut absolument pas qu'elle ait des effets cachés, ce qu'on appelle des "effets de bords". Par exemple, si vous imaginez la fonction racine carrée qui calcule la racine d'un nombre, son prototype typiquement, comme on peut le trouver dans la bibliothèque standard, c'est de recevoir un argument de type "double x", elle s'appelle sqrt, et elle retourne un double, cette fonction, vous vous attendez à ce qu'elle calcule uniquement la racine carrée, et non pas à ce qu'elle pollue l'affichage avec des messages. Donc, c'est ce qu'on appelle des effets de bords. C'est quand on a des effets qui ne sont pas attendus. Une fois qu'on a bien spécifié ce que doit faire la fonction, alors il faut bien sûr choisir un nom pertinent qui illustre exactement ce que fait la fonction. Enfin, un dernier conseil, commencer par toujours faire le prototype de votre fonction. Faire le prototype de votre fonction, ça va vous permettre de clarifier ce qu'elle doit recevoir, les paramètres, on l'a vu tout à l'heure

notes

résumé

0m 1s

Résumé : attention à la syntaxe !

EPFL

```
int a;      : déclaration de variable non initialisée
int a();    : prototype de fonction sans paramètre
int a(5);   : déclaration/initialisation de variable
a(5);       : appel de fonction à un argument
```

et aussi ce qu'elle doit fournir : la valeur de retour, le type de retour. Ce seront donc déjà deux choses que vous aurez spécifiées avant même de vous préoccuper de savoir comment vous allez faire ce que doit faire la fonction. Résumons maintenant différents aspects de la syntaxe que nous avons vu jusqu'ici dans ce cours. Si vous écrivez une expression qui commence par un type,

notes

résumé

1m 13s

Résumé : attention à la syntaxe !

EPFL

```
int a;      : déclaration de variable non initialisée
int a();    : prototype de fonction sans paramètre
int a(5);   : déclaration/initialisation de variable
a(5);       : appel de fonction à un argument
```

int f();
int demander_nb();

ici suivi d'un nom avec un point virgule, alors ce sera la déclaration d'une variable. Par contre, si vous ajoutez entre le nom et le point virgule, ici des parenthèses sans argument, ça sera le prototype d'une fonction, et non pas une variable. Cela peut vous paraître peut-être un petit peu plus naturel si je l'avais écrit de cette façon ici, int f(); on voit ici, peut-être à cause du nom, mieux que c'est une fonction. Ou alors, encore mieux, si je vous écris, comme je l'ai fait dans l'exemple précédent la fonction "demande nb", ici qui ne prend aucun argument. Faites très attention, ici c'est bien un prototype de fonction. Et ça n'a rien à voir avec une variable, c'est une erreur fréquemment commise. Si par contre, entre les parenthèses, vous mettez une valeur qui est compatible avec le type

notes

résumé

1m 37s

Résumé : attention à la syntaxe !

EPFL

```
int a;      : déclaration de variable non initialisée
int a();    : prototype de fonction sans paramètre
int a(5);   : déclaration/initialisation de variable
a(5);       : appel de fonction à un argument
```

int f();
int demander_nb();

z = a

qui est au début de la ligne en question, alors vous avez de nouveau une déclaration de variable et cette fois-ci, avec la valeur, c'est une initialisation. Donc cette ligne est en fait très proche de la première ligne. Simplement on rajoute ici une valeur pour une initialisation. Et ça n'a absolument rien à voir avec la deuxième ligne. Donc faites très attention à ce genre d'écriture. Enfin, pour l'appel de la fonction, si on met un nom sans type devant, il n'y a absolument pas de type devant,

notes

résumé

2m 25s

ou on peut avoir une expression de type = devant, par exemple z=a(5) mais il n'y a pas de type ici alors c'est bien sûr un appel de fonction ça vous aurait peut-être paru plus naturel si j'avais écrit z=f(5) là vous reconnaissez naturellement un appel à une fonction f ou si j'avais écrit z=sqrt(2.0) Là, vous reconnaissez l'appel d'une fonction. l'appel d'une fonction.

notes

résumé

2m 53s