



Langage C - Les fonctions

Les fonctions



Langage C

Langage C
Caractéristiques
Préprocesseur
Types de données
Les variables
Les opérateurs
Structures conditionnelles
Les fonctions
Les tableaux
Les structures
Les pointeurs
Chaîne de caractères
Listes chaînées

La notion de fonction

On appelle *fonction* un sous-programme qui permet d'effectuer un ensemble d'instructions par simple appel de la fonction dans le **corps** du programme principal. Les fonctions permettent d'exécuter dans plusieurs parties du programme une série d'instructions, cela permet une simplicité du code et donc une taille de programme minimale. D'autre part, une fonction peut faire appel à elle-même, on parle alors de fonction récursive (il ne faut pas oublier de mettre une condition de sortie au risque sinon de ne pas pouvoir arrêter le programme...).

La déclaration d'une fonction

Avant d'être utilisée, une fonction doit être définie car pour l'appeler dans le corps du programme il faut que le compilateur la connaisse, c'est-à-dire qu'il connaisse son nom, ses arguments et les instructions qu'elle contient. La définition d'une fonction s'appelle « *déclaration* ». La déclaration d'une fonction se fait selon la syntaxe suivante :

```
type_de_donnee Nom_De_La_Fonction(type1 argument1, type2 argument2, ...) {
```

```
liste d'instructions
```

```
}
```

Remarques :

- type_de_donnee représente le type de valeur que la fonction est sensée retourner (char, int, float...)
- Si la fonction ne renvoie aucune valeur, on la fait alors précéder du mot-clé *void*
- Si aucun type de donnée n'est précisé (cela est très vilain !), le type *int* est pris par défaut
- Le nom de la fonction suit les mêmes règles que les **noms de variables** :
 - le nom doit commencer par une lettre
 - un nom de fonction peut comporter des lettres, des chiffres et les caractères `_` et `&` (les espaces ne sont pas autorisés !)
 - le nom de la fonction, comme celui des variables est sensible à la casse (différenciation entre les minuscules et majuscules)
- Les arguments sont facultatifs, mais s'il n'y a pas d'arguments, les parenthèses doivent rester présentes
- Il ne faut pas oublier de refermer les accolades



- Le nombre d'accolades ouvertes (fonction, boucles et autres structures) doit être égal au nombre d'accolades fermées !
- La même chose s'applique pour les parenthèses, les crochets ou les guillemets !

Une fois cette étape franchie, votre fonction ne s'exécutera pas tant que l'on ne fait pas appel à elle quelque part dans la page !

Appel de fonction

Pour exécuter une fonction, il suffit de faire appel à elle en écrivant son nom (une fois de plus en respectant la casse) suivi d'une parenthèse ouverte (éventuellement des arguments) puis d'une parenthèse fermée :

```
Nom_De_La_Fonction();
```

Remarques :

- le point-virgule signifie la fin d'une instruction et permet au navigateur de distinguer les différents blocs d'instructions
- si jamais vous avez défini des arguments dans la déclaration de la fonction, il faudra veiller à les inclure lors de l'appel de la fonction (le même nombre d'arguments séparés par des virgules !)

```
Nom_De_La_Fonction(argument1, argument2);
```

Prototype d'une fonction

Le prototype d'une fonction est une description d'une fonction qui est définie plus loin dans le programme. On place donc le prototype en début de programme (avant la fonction principale *main()*).

Cette description permet au compilateur de « vérifier » la validité de la fonction à chaque fois qu'il la rencontre dans le programme, en lui indiquant :

- Le type de valeur renvoyée par la fonction
- Le nom de la fonction
- Les types d'arguments

Contrairement à la **définition** de la fonction, le prototype n'est pas suivi du corps de la fonction (contenant les instructions à exécuter), et ne comprend pas le nom des paramètres (seulement leur type).

Un prototype de fonction ressemble donc à ceci :

```
Type_de_donnee_renvoyee Nom_De_La_Fonction(type_argument1, type_argument2, ...);
```



Le prototype est une instruction, il est donc suivi d'un point-virgule !

Voici quelques exemples de prototypes :

```
void Affiche_car(char, int);

int Somme(int, int);
```

Les arguments d'une fonction

Il est possible de passer des arguments à une fonction, c'est-à-dire lui fournir une valeur ou le nom d'une variable afin que la fonction puisse effectuer des opérations sur ces arguments ou bien grâce à ces arguments.
Le passage d'arguments à une fonction se fait au moyen d'une liste d'arguments (séparés par des virgules) entre parenthèses suivant immédiatement le nom de la fonction.
Le nombre et le type d'arguments dans la déclaration, le prototype et dans l'appel doit correspondre au risque, sinon, de générer une erreur lors de la compilation...

Un argument peut être :

- une constante
- une variable
- une expression
- une autre fonction

Renvoi d'une valeur par une fonction

La fonction peut renvoyer une valeur (et donc se terminer) grâce au mot-clé *return*. Lorsque l'instruction *return* est rencontrée, la fonction évalue la valeur qui la suit, puis la renvoie au programme appelant (programme à partir duquel la fonction a été appelée).
Une fonction peut contenir plusieurs instructions *return*, ce sera toutefois la première instruction *return* rencontrée qui provoquera la fin de la fonction et le renvoi de la valeur qui la suit.
La syntaxe de l'instruction *return* est simple :

```
return (valeur_ou_variable);
```







Le type de valeur retourné doit correspondre à celui qui a été précisé dans la définition (et le prototype).

Trucs & astuces pertinents trouvés dans la base de connaissances

- 04/11 15h50
- 20/02 11h25
- 09/01 20h26
- 30/11 22h37
- 18/11 12h04
- 18/11 01h10
- 15/11 02h50
- 21/09 09h44
- 27/06 23h31
- 21/01 03h06
- Rediriger en fonction de la langue du visiteur (PHP)
- Mon accès à internet ne fonctionne plus (Connexion)
- C/C++ Erreur de segmentation (Langage C)
- Comment débiter, quel langage? (Langages)
- Toutes les fonctions de google sur un seul site (Google)
- Fonction mail() (PHP)
- Langage informatique=Langage de programmation (Mythes et légendes)
- Critères de choix d'un langage/framework (Programmation)
- Fonction définition (Moteurs de recherche)
- Envoyer un mail avec pièce jointe (Langages)

Plus d'astuces sur « Langage C fonctions »

Discussions pertinentes trouvées dans le forum

27/02 19h09	 fonction MUL de l'assembleur en langage C	Développement	27/02 23h28->slolo2000	3
09/11 23h11	 aide appel de fonctions (langage C)	Développement	10/11 10h49->LeSousss	6
03/12 13h48	 langage c comment fonction gets??	Développement	04/12 13h14->Guki	6
14/12 00h07	 Langage C / Fonction récursive	Développement	24/12 15h02->Belhauss	5
23/05 17h33	 langage c et fonction	Développement	24/05 13h21->charly	4
12/12 12h10	 Fonction en langage C.	Développement	15/12 10h36->Bravi	4
12/07 18h19	 fonction en langage C pour obtenir l'adre IP	Développement	16/07 09h49->batmat	3
30/06 00h49	 [fonction en langageC]	Développement	30/06 17h21->Char Snipeur	3
08/02 19h45	 langage C: fonction :aide	Développement	26/03 20h40->fonceurweb	2
30/03 07h44	 Documentation des fonctions du langage C	Développement	30/03 08h01->Psyko	2
 Discussion fermée  Problème résolu		Plus de discussions sur « Langage C fonctions »		

Ce document intitulé « Langage C - Les fonctions » issu de l'encyclopédie informatique Comment Ça Marche (www.commentcamarche.net) est mis à disposition sous les termes de la licence Creative Commons. Vous pouvez copier, modifier des copies de cette page, dans les conditions fixées par la licence, tant que cette note apparaît clairement.

