# Programmation C++ (débutant)/Les fonctions

## Le cours sur les fonctions

### Pourquoi écrire des fonctions

Lorsqu'on a un ensemble de lignes de code qui doivent être exécutées à différents endroits dans un programme, au lieu de réécrire les mêmes lignes de code, il est intéressant de créer des fonctions.
Au lieu d'écrire une fonction main() de 500 lignes, il est préférable de créer 25 fonctions de 20 lignes

- on structure le programme.
- il est plus facile de tester chaque fonction.

Il est impossible d'avoir en tête plus de 1000 lignes de codes : or la plupart des programmes réels comportent des dizaines de milliers de lignes et les grosses applications en comportent des millions. Ecrire des fonctions est absolument obligatoire.

### Ecrire une fonction

**Syntaxe :** type identificateur(paramètres) { Corps de la fonction }
A chaque appel de la fonction on exécute le corps de la fonction.
L'identificateur est le nom de la fonction.
La fonction peut avoir des paramètres.
La fonction peut renvoyer une valeur de type type.

### Appel à une fonction

Lors de l'appel de la fonction, le programme exécute la totalité des instructions du corps de la fonction, puis reprend le programme juste après l'appel de la fonction.

### Exemple 1 : un exemple de fonction

```
#include <iostream>
using namespace std;

void b()
{
    cout<<"Bonjour"<<endl;
}

int main()
{
    cout<<"COUCOU1"<<endl;
    b();
    cout<<"COUCOU2"<<endl;
    b();
    b();
    cout<<"COUCOU3"<<endl;
    b();
    return 0;
}
```

- **Explications**
  - Dans ce programme, on a créé une fonction b qui se contente d'afficher "Bonjour" à l'écran. La fonction b est précédée du type void : cela signifie que la fonction ne renvoie aucune valeur au programme appelant.
  - Le programme principal (la fonction main()) affiche "COUCOU1" à l'écran, ensuite appelle la fonction b, affiche le message "COUCOU2" à l'écran, appelle ensuite 2 fois la fonction b, affiche le message "COUCOU3" et appelle une dernière fois la fonction b.
- **Exécution de l'exemple 1**

Lorsqu'on exécute le programme voici ce qu'on obtient à l'écran :

```
COUCOU1
Bonjour
COUCOU2
Bonjour
Bonjour
COUCOU3
Bonjour
```

## Les paramètres

On peut paramétrer une fonction : les paramètres permettent de rendre la fonction plus générale donc plus facilement réutilisable. La réutilisation de code est une des notions fondamentales du langage C++. La notion de fonction est une des premières méthodes nous permettant d'aborder le sujet.

## Exemple 2 : une fonction avec des paramètres

```
#include <iostream>
using namespace std;

void b(int i)
{
    int j;
    for(j=0; j<i; j++) cout<<"Bonjour"<<endl;
}

int main()
{
    cout<<"COUCOU1"<<endl;
    b(2);
    cout<<"COUCOU2"<<endl;
    b(3);
    return 0;
}
```

- **Explications**
  - Cette fois-ci la fonction b comporte un paramètre entier i. Cette fonction affiche i fois "Bonjour" à l'écran.
  - Dans le programme principal, la fonction main() affiche "COUCOU1", appelle ensuite la fonction b avec comme paramètre 2, **Lors de l'appel b(2) de la fonction, on recopie la valeur 2 dans i puis on exécute le corps de la fonction avec cette valeur de i : on affiche donc 2 fois "Bonjour".

- Lors de l'appel b(3) de la fonction, on recopie la valeur 3 dans i puis on exécute le corps de la fonction avec cette valeur de i : on affiche donc 3 fois "Bonjour".

- **Exécution**

Lorsqu'on exécute le programme voici ce qu'on obtient à l'écran :

```
COUCOU1
Bonjour
Bonjour
COUCOU2
Bonjour
Bonjour
Bonjour
```

## Environnement d'une fonction

- Une fonction peut avoir ses propres variables locales.
- On appelle environnement d'une fonction l'ensemble des variables accessibles dans le corps de la fonction.
- Il est possible de définir des variables globales (voir la définition des constantes) : il est fortement déconseillé d'utiliser des variables globales autres que des constantes.
- L'environnement d'une fonction comprend :
  - les variables locales à la fonction.
  - les paramètres de la fonction.
  - les variables globales.

## Une fonction qui renvoie une valeur

Une fonction peut renvoyer une valeur d'un certain type à l'environnement appelant.
**void** signifie que la fonction ne renvoie rien. Si on veut que la fonction renvoie une certaine valeur, il faudra écrire un autre type à la place de void.
Le mot clé return permet :

- d'arrêter l'exécution de la fonction (il est en général placé à la fin de la fonction)
- de renvoyer une valeur à l'environnement appelant.

Si une fonction b renvoie un entier, l'appel s'écrira : a=b(…); a permettra de récupérer la valeur renvoyée par le return.

## Exemple 3 : une fonction qui renvoie une valeur

```
#include<iostream>
using namespace std;

int b(int i, int j)
{


    int k;
    k = i*i + j*j;
    return k;
}
```

```
int main()
{
    int a;
    a = b(3,4);
    cout<<"Le resultat vaut : "<<a<<endl;
    return 0;
}
```

- **Explications**

Lors de l'appel **a=b(3,4);**

- 3 et 4 sont respectivement recopiés dans les variables i et j.
- On exécute le corps de la fonction (qui contient une variable locale k)
- **return k** permet d'arrêter l'exécution de la fonction en renvoyant la valeur de k qui est recopiée dans a.
- L'environnement de la fonction main ne comprend que la variable a : on ne peut pas utiliser les variables i, j et k dans la fonction main.
- L'environnement de la fonction b comprend les variables i,j et k : on ne peut pas utiliser a dans la fonction b.
- La séparation des environnements permet de mieux structurer les applications.

- **Exécution :**

Lorsqu'on exécute le programme voici ce qu'on obtient à l'écran :
Le resultat vaut : 25

### Exemple 4 : une autre fonction

```
#include <iostream>
using namespace std;

int b(int i)
{
    int k, s=0;
    for(k=1; k<=i; k++) s = s+k*k;
    return s;
}

int main()
{
    int a;
    a = b(4);
    cout<<"Le résultat vaut :"<<a<<endl;
    return 0;
}
```

- **Explications**
  - On définit une fonction b qui a un paramètre entier i et qui calcule la somme des carrés des i premiers entiers.
  - Dans la fonction main, on appelle b(4) et on récupère la valeur renvoye dans la variable a. a contient donc la somme des 4 premiers carrés.
- **Exécution**

Lorsqu'on exécute le programme voici ce qu'on obtient à l'écran :
Le résultat vaut : 30

### Exemple 5 : une fonction manipulant des réels

```
#include<iostream>
using namespace std;

double b(double x, double y)
{
    double m;
    m = (x+y) / 2;
    return m;
}

int main()
{
    double a;
    a = b(3.2, 4.2);
    cout<<"Le résultat vaut :"<<a<<endl;
    return 0;
}
```

- **Explications :**
  - Dans cette exemple, la fonction b a 2 paramètres de type double, nommées x et y. La fonction b renvoie un double. Cette fonction renvoie la moyenne de x et de y. La variable m est une variable locale à la fonction b.
  - Dans le programme principal, on récupère dans a la moyenne de 3.2 et 4.2 en appelant la fonction b.
- **Exécution :**

Lorsqu'on exécute le programme voici ce qu'on obtient à l'écran :
Le résultat vaut : 3.7

### Compatibilité des types

Une fonction peut avoir des paramètres de différents types. Lors de l'appel, il convient de respecter l'ordre et le type des paramètres entre l'entête et le corps de la fonction.

### Passage de tableaux en paramètres

Lorsqu'on passe un tableau (d'entiers par exemple en paramètre), il y a identification entre le tableau de l'environnement appelant et le paramètre de la fonction. Toute modification du tableau dans la fonction est répercutée dans le tableau de l'environnement appelant.

### Exemple 6 : passage de tableaux en paramètre

```
#include <iostream>
using namespace std;

const int n=4;

void saisir(int t[n])
{
    int i;
    for(i=0; i<n; i++)
```

```
    {
        cout<<"Tapez la valeur numero "<<i<<" : ";
        cin >> t[i];
    }
}

void affiche(int t[n])
{
    int i;
    for(i=0; i<n; i++) cout<<"La valeur numero "<<i<<" est : "<<t[i]<<endl;
}

int main()
{
    int a[n];
    saisir(a);
    affiche(a);
    return 0;
}
```

- **Explications**
  - Lors de l'appel saisir(a), il y a identification du tableau a et du paramètre t de la fonction saisir : toute modification de t modifie le tableau a. La fonction saisir a permet de demander à l'utilisateur de saisir une à une toutes les cases d'un tableau de n cases.
  - La fonction affiche à un tableau de n entiers en paramètres et affiche toutes les cases de ce tableau.
  - Le programme est structuré : il est constitué d'un ensemble de fonctions courtes dont le rôle peut être facilement identifié.
  - Le main() devient un programme très court !
- **Exécution**

Lorsqu'on exécute le programme voici ce qu'on obtient à l'écran :

```
Tapez la valeur numero 0 : 4
Tapez la valeur numero 1 : 6
Tapez la valeur numero 2 : 5
Tapez la valeur numero 3 : 7
La valeur numero 0 est : 4
La valeur numero 1 est : 6
La valeur numero 2 est : 5
La valeur numero 3 est : 7
```

### Attention

Une modification d'un paramètre qui n'est pas un tableau dans le corps d'une fonction n'est pas répercuté dans l'environnement appelant.

### Exemple 7

```
#include <iostream>
using namespace std;

void saisir(int n)
{
    cout<<"Tapez un entier : "; cin>>n;
}

void affiche(int n)
{
    cout<<"La valeur de l'entier est : "<<n<<endl;
}

int main()
{
    int x;
    x=54;
    saisir(x);
    affiche(x);
    return 0;
}
```

- **Explications**
  - Dans la fonction main, on définit une variable entière x et on l'initialise à 54.
  - On appelle ensuite la fonction saisir : on copie la valeur de x dans n. Dans le corps de la fonction saisir, on demande à l'utilisateur de taper une valeur qui est mise dans x. Mais à aucun moment cette valeur x n'est mise dans la variable x : la valeur tapée par l'utilisateur est donc perdue.
  - La valeur de x reste donc à 54. Lorsqu'on appelle la fonction affiche, on affiche la valeur 54.
- **Exécution**

Lorsqu'on exécute le programme voici ce qu'on obtient à l'écran :
Tapez un entier : **20**
La valeur de l'entier : 54

### Passage de paramètres par référence

On peut passer un paramètre par référence (et non par copie) en indiquant dans l'entête de la fonction un & après le type. Il y a alors identification du paramètre de la fonction et de la variable de l'environnement appelant.

### Exemple 8 : passage de paramètres par référence

```
#include <iostream>
using namespace std;

void saisir(int & n)
{
    cout<<"Tapez un entier : "; cin>>n;
}

void affiche(int n)
{
    cout<<"La valeur de l'entier : "<<n<<endl;
}

int main()
{
    int x;
    x=54;
    saisir(x);
    affiche(x);
    return 0;
}
```

- **Explications**
  - Lors de l'appel saisir(x), il y a identification des variables x et n : toute modification de n modifie la valeur de x. Lorsque l'utilisateur saisit la valeur de n dans l'instruction cin>>n; , il y a modification du contenu de la variable x. On récupère dans x la valeur tapée par l'utilisateur
  - On appelle ensuite la fonction affiche qui affiche la valeur de x.
- **Exécution**

Lorsqu'on exécute le programme voici ce qu'on obtient à l'écran :
Tapez un entier : **20**
La valeur de l'entier: 20

### Paramètres en entrées et en sorties

- Certains paramètres vont fournir des données à une fonction : les paramètres en entrée.
- D'autres vont permettre de renvoyer une valeur vers l'environnement appelant : les paramètres en sortie.
- Techniquement , on utilisera le return ou le passage de paramètres par référence pour envoyer une valeur à l'environnement appelant.
- D'autres peuvent éventuellement avoir les 2 rôles : les paramètres en entrée/sortie.

### Exemple 9 : paramètres en entrées et en sorties

```
#include <iostream>
using namespace std;


void minmax(int i, int j, int & min, int & max)
{
    if(i<j) {min=i; max=j;} else {min=j; max=i;};
}

int main()
{
    int a,b,w, x;

    cout << "Tapez la valeur de a : "; cin >> a;
    cout << "Tapez la valeur de b : "; cin >> b;
    minmax(a,b,w,x);
    cout << "Le plus petit vaut : " << w << endl;
    cout << "Le plus grand vaut : " << x << endl;
    return 0;
}
```

- **Explications**
  - La fonction minmax permet de récupérer le plus petit et le plus grand de 2 entiers i et j.
  - Dans la fonction minmax, i et j sont les paramètres en entrée et min et max sont les paramètres en sorties.
  - Dans le main, lorsqu'on appelle la fonction minmax, on recopie les valeurs de a et de b respectivement dans i et j et il y a identification des variables w et min et des variables x et max durant toute la durée de l'appel de la fonction.
  - Dans w on récupère donc le plus petit de a et de b et dans x le plus grand de a et de b.
- **Exécution**

Lorsqu'on exécute le programme voici ce qu'on obtient à l'écran :
Tapez la valeur de a : **15**
La valeur de b est : **11**
Le plus petit vaut 11
Le plus grand vaut 15

### Passer un tableau de taille quelconque

- Le paramètre d'une fonction peut être un tableau de taille quelconque.
- Dans l'environnement appelant, la taille du tableau devra être déterminée.
- En général, lorsqu'on passe un tableau de taille quelconque, on passe un autre paramètre entier qui indique la taille du tableau.

### Exemple 10 : passer un tableau de taille quelconque

```
#include <iostream>
using namespace std;

void saisir(int t[], int n)
{
    int i;
    for(i=0; i<n; i++)
    {
        cout << "Tapez l'entier numero " << i << " : ";
        cin >> t[i];
    }
}

void afficher(int t[], int n)
{
    int i;
    for(i=0; i<n; i++)
        cout << "L'entier numero " << i << " vaut : " << t[i] << endl;
}

int main()
{
    int a[3], b[5];
    cout << "SAISIE DU TABLEAU a" << endl;
    saisir(a, 3);
    cout << "SAISIE DU TABLEAU b" << endl;
    saisir(b, 5);
    cout << "AFFICHAGE DU TABLEAU a" << endl;
    afficher(a, 3);
    cout << "AFFICHAGE DU TABLEAU b" << endl;
    afficher(b, 5);
    return 0;
}
```

- **Explications**
  - Dans la fonction saisir, le premier paramètre est int t[] : il s'agit donc d'un tableau d'entiers de n'importe quelle taille. Le deuxième paramètre n indique la taille du tableau. Cette fonction demande à l'utilisateur de saisir une à une toutes les cases du tableau.
  - La fonction affiche a également comme paramètres un tableau d'entiers de taille quelconque et un paramètre n qui indique la taille du tableau. Cette fonction affiche le contenu du tableau.

- Dans le programme principal, on saisit le contenu du tableau a contenant 3 cases et du tableau b contenant 5 cases en ** On constate toutefois que dans la fonction main, la taille des tableaux a et b est belle et bien connue.

- **Exécution**

Lorsqu'on exécute le programme voici ce qu'on obtient à l'écran :

```
SAISIE DU TABLEAU a
Tapez l'entier numero 0: '''123'''
Tapez l'entier numero 1: '''456'''
Tapez l'entier numero 2: '''789'''
SAISIE DU TABLEAU b
Tapez l'entier numero 0: '''987'''
Tapez l'entier numero 1: '''654'''
Tapez l'entier numero 2: '''321'''
Tapez l'entier numero 3: '''741'''
Tapez l'entier numero 4: '''852'''
AFFICHAGE DU TABLEAU a
L'entier 0 vaut : 123
L'entier 1 vaut : 456
L'entier 2 vaut : 789
AFFICHAGE DU TABLEAU b
L'entier 0 vaut : 987
L'entier 1 vaut : 654
L'entier 2 vaut : 321
L'entier 3 vaut : 741
L'entier 4 vaut : 852
```

## Portée des variables

- Dans le main, on peut accéder :
  - aux variables locales du main
  - aux constantes globales
- Dans une fonction, on peut accéder
  - aux paramètres de la fonction
  - aux constantes globales
- Il est déconseillé de définir des variables globales non constantes !
- Les échanges entre les environnements se font :
  - par les returns
  - par les passages de paramètres par référence

### Conclusion

- Il faudra maîtriser les passages de paramètres par valeur et par référence ainsi que le rôle particulier des tableaux.
- Il est indispensable d'avoir bien compris le rôle des paramètres en entrée et en sortie d'une fonction.
- Les fonctions vont jouer un rôle fondamental dans la structuration de nos programmes. Nos programmes vont maintenant être constitués d'une multitude de fonctions qui auront une taille petite (10 à 30 lignes) et qui auront toutes un rôle très précis.
- Cette méthodologie est indispensable pour créer de longs programmes.

## Exercices sur les fonctions

### EXERCICE 1

Ecrire une fonction distance ayant comme paramètres 4 doubles xa,ya et xb,yb qui réprésentent les coordonnées de deux points A et B et qui renvoie la distance AB. Tester cette fonction.

### EXERCICE 2

Ecrire une fonction f ayant comme paramètres un double x et un booléen ok et qui renvoie un double par un return. La fonction renvoie par un return la racine carrée de (x-1)*(2-x). La fonction renvoie par l'intermédiaire de la variable ok la valeur true si la fonction est définie au point x, false sinon. Tester cette fonction.

### EXERCICE 3

Ecrire une fonction f ayant en paramètre un entier et qui renvoie par un return un booléen : true si l'entier est premier et false sinon. Tester cette fonction.

### EXERCICE 4

Ecrire une fonction f ayant comme paramètre un entier n et qui renvoie le n-ième nombre premier : cette fonction utilisera la fonction du 3). Tester cette fonction.

### EXERCICE 5

Ecrire une fonction swap ayant en paramètres 2 entiers a et b et qui echange les contenus de a et de b. Tester cette fonction.

### EXERCICE 6

Ecrire une fonction f ayant en paramètres un tableau t de taille quelconque et un entier n indiquant la taille du tableau. f doit renvoyer par un return un booleén b indiquant s'il existe une valeur comprise entre 0 et 10 dans les n premières cases du tableau t. Tester cette fonction.

### EXERCICE 7

Ecrire une fonction f ayant en paramètres un tableau t de taille quelconque et un entier n indiquant la taille du tableau. f doit renvoyer par un return le nombre de valeurs comprises entre 0 et 10 dans les n premières cases du tableau t.Tester cette fonction.

## EXERCICE 8

Ecrire une fonction f ayant en paramètres un tableau t de taille quelconque et un entier n indiquant la taille du tableau. f possède un autre paramètre v , entier passé par référence. f doit renvoyer par un return un booleén b indiquant s'il existe une valeur comprise entre 0 et 10 dans les n premières cases du tableau t. Si f renvoie true, v est égal à la valeur de la première case du tableau comprise entre 0 et 10. Tester cette fonction.

## EXERCICE 9

Ecrire une fonction f ayant en paramètres un tableau t1 de taille quelconque et un entier n indiquant la taille du tableau, ainsi qu'un tableau t2 de la même taille que t1. f doit renvoyer par un return un entier nb indiquant le nombre de valeurs comprises entre 0 et 10 dans le tableau t1. f doit mettre dans le tableau t2 les différentes valeurs comprise entre 0 et 10 qu'il a rencontrées dans le tableau t1.

## EXERCICE 10

Ecrire une fonction f ayant en paramètres un tableau t de taille quelconque et un entier n indiquant la taille du tableau. f doit renvoyer par un return un entier égal à l'indice de la première case du tableau (parmi les n premières) comprise entre 0 et 10. S'il n'existe pas de telle valeur, la fonction renvoie -1. Tester cette fonction.

# Sources et contributeurs de l'article

**Programmation C++ (débutant)/Les fonctions** *Source*: http://fr.wikibooks.org/w/index.php?oldid=335320 *Contributeurs*: Adniang75, JackPotte, Merrheim, Tavernier, Trefleur, 28 modifications anonymes

# Licence