AIDE-MÉMOIRE

C++

Christophe **Pichaud**

AIDE-MÉMOIRE C++

DUNOD

Mise en page : Belle Page

© Dunod, 2020
11 rue Paul Bert, 92240 Malakoff
www.dunod.com
ISBN 978-2-10-080712-3

Table des matières

Ce livre est dédié à mes parents Jean-Marc et Mireille, qui ont toujours cru en moi et m'ont toujours encouragé.

Le support de mes filles Edith (Didou), Lisa (le mini-mini) et Audrey (Maggie) a aussi été important.

Jérome (petit Lulu), je ne sais pas si on va partir aux US avec ce livre mais on y travaille…

Préface

Pourquoi apprendre le C++ en 2020 ? Avant tout, en raison de son omniprésence : le C++ est disponible sur tous les matériels – des périphériques embarqués aux super calculateurs – et sur toutes les plateformes. C'est le langage dans lequel sont développés les systèmes d'exploitation mais aussi les grandes applications comme Adobe Photoshop ou Microsoft Office mais aussi sur Facebook ou Google. Directement ou indirectement *via* les moteurs 3D, c'est également lui qui fait fonctionner tous les jeux modernes. Le C++ est un langage vivant : trouvant ses racines au début des années 1970 avec le C des fameux Kerninghan et Richie, il a su évoluer pour devenir un langage moderne, portable et standardisé, tout en conservant ses performances incontestées. Le C++ a aussi de nombreuses vertus pédagogiques : programmation orientée objets, généricité, algorithmes, gestion automatique de la mémoire, fonctions lambdas, tout y est !

Et qui de mieux que Christophe pour nous accompagner dans cet apprentissage ? Depuis de nombreuses années, il n'a de cesse de partager sa passion pour la programmation et le C++ en particulier. Qu'il s'agisse de conseils auprès des entreprises, d'enseignements en école d'ingénieurs, de présentations techniques lors de grands événements ou encore de nombreux articles dans la presse-spécialisée ou sur son blog, l'énergie de Christophe n'a pas de limites.

Ce livre s'adresse à la fois aux jeunes développeurs curieux d'appendre ce langage roi et à ceux, peut-être un peu moins jeunes, qui l'ont pratiqué dans le passé, avant sa grande cure de modernisation concrétisée par le C++11 et les mises à jour régulières qui ont suivi. Il commence par explorer les bases du langage et ses racines communes avec le C, puis introduit les classes et surcharges d'opérateurs pour une programmation orientée objets

et poursuit avec l'utilisation des templates qui accroissent encore les capacités d'expression du langage. La deuxième partie du livre passe en revue les principaux concepts de la bibliothèque standard STL : celle-ci assure à la fois la portabilité du code et la productivité du développeur.

Pour que cet ouvrage complet soit plus qu'une documentation de référence, Christophe marie un style dynamique et une grande pédagogie et n'oublie pas de distiller des bonnes pratiques de programmation au fur et à mesure de l'introduction des différents concepts. Bienvenue dans la programmation du XXIe siècle !

Alain Zanchetta

Principal SDE sur Microsoft HoloLens

Avant-propos

The World is built on C++.

Herb Sutter

C++ est un langage profondément riche et intense. Avec le temps, il est devenu élégant et moderne. Malgré des milliards de dollars dépensés en marketing, il n'a pas été remplacé par les langages dits « productifs », les systèmes d'exploitation comme Microsoft Windows, Mac OS ou Linux, les suites bureautiques comme Office ou les jeux vidéo étant toujours faits avec.

En 2018, j'ai contacté Dunod pour leur proposer ce projet de livre car il n'existe que peu de livres sur le C++ moderne, c'est-à-dire le C++ post C++11, date à laquelle un ensemble de fonctionnalités comme les pointeurs intelligents, la déduction de type automatique (auto) et la suprématie de la bibliothèque standard STL avec ses strings, containers et algorithmes entre autres sont apparues.

Beaucoup de livres C+ en français parlent de C++03 mais ne détaillent pas les fonctionnalités des derniers standards comme C++14 et C++17. Le nouveau standard C++20 est arrivé en janvier 2020. Ce livre vous permet, par une approche par l'exemple, de connaître toutes ses fonctionnalités.

En tant que professionnel du développement C++, je suis MVP[1] *Developer Technologies* et j'ai accès au code source des derniers OS Windows. Pour vous donner une idée rapide, Microsoft a toujours fait ses OS et ses logiciels

1 *Most Valuable Professional. Cf.* https://mvp.microsoft.com/.

en C/C++. Pour Microsoft, le C++ est naturel, c'est comme l'électricité. Il n'y a pas de débat. On veut développer un produit, on le fait en C++. C'est avant tout une nécessité de performance par défaut et de ce que les gens nomment « Power and Performance ».

Le C++ ne fait aucun compromis là-dessus. Sans violer le NDA[1] qui me lie à Microsoft, je peux vous dire que le code source de Windows 10 est truffé de C++ moderne, des templates, d'utilisation de la STL et utilise toutes les fonctionnalités possibles de C++. Le code est propre, dense, orienté objet et, comme le dit Marian Luparu, Principal Group Manager de Visual C++, « *Microsoft is committed to C++* » (Microsoft s'engage sur C++). Il ne faut pas oublier que son équipe et son produit – Visual C++ (MSVC), le compilateur C++ de Microsoft – compilent les produits phares comme Windows 10, Windows Server, Office, SQL Server entre autres. Visual C++ est donc le joyau interne de Microsoft : c'est lui qui fabrique les logiciels phares de la société de Redmond. Chez Microsoft, chaque groupe produit fait du C++.

En 2011, Channel9 sortait une vidéo de Craig Symonds et Mohsen Agsen intitulée *C++ Renaissance*[2]. Cette renaissance signifie que, par définition, le langage, les compilateurs et l'outillage de composition évoluent vers un état qui maximise l'efficacité, la productivité et la créativité des développeurs natifs dans tous les domaines matériels et logiciels (PC, appareils mobiles, systèmes d'exploitation, applications utilisateur, services, etc.). C++ est un langage de programmation « système » puissant, mais c'est plus que cela.

Dans le monde Linux, la suprématie de C/C++ est sans comparaison. Comme dans Windows, le noyau (kernel) et les pilotes de périphériques (drivers) sont écrits en C et tout le reste en C++ sauf les couches historiques. Les applications phares comme Apache, MySQL, Redis pour ne citer qu'elles sont faites en C/C++. Il existe même une distribution Linux nommée Gentoo qui propose de construire son environnement Linux à partir des sources. Le système *portage* télécharge un petit compilateur pour compiler GCC puis GCC compile tout : le noyau, les drivers, les couches de l'OS, Bash & Tools, X-Motif, Gnome, KDE, etc.

1 *Non Disclosure Agreement* ou Accord de confidentialité.

2 https://channel9.msdn.com/Shows/Going+Deep/Craig-Symonds-and-Mohsen-Agsen-C-Renaissance.

Pourquoi lire ce livre ?

Le C++ moderne est en pleine expansion. On le retrouve sur :

- les objets connectés (IoT) ;
- les smartphones ;
- les postes de travail ;
- les serveurs ;
- le cloud et l'intelligence artificielle (IA).

C++ est le langage le plus puissant et le plus rapide qui existe. Il tire parti au maximum de l'architecture matérielle et des processeurs avec une empreinte mémoire limitée.

Dans ce livre, le C++ moderne est expliqué à travers différents exemples. Le lecteur est tout de suite confronté à string, vector, aux algorithmes et il n'est pas nécessaire de connaître le C pour apprivoiser C++. L'utilisation de la bibliothèque standard STL (*Standard Template Library*) est une approche moderne du C++.

Structure du livre

Le livre est organisé en deux parties :

- partie I : Le langage C++ ;
- partie II : La bibliothèque standard (STL).

La partie I contient les chapitres sur les fondamentaux, les variables et les types, strings, vecteurs, Itérateur, Array, Le programme de démo, les expressions, les déclarations, les fonctions, les classes, les opérations de copie et de déplacement, les opérations de surcharge, la programmation orientée objet et les templates.

La partie II traite les chapitres sur les utilitaires, les opérations numériques et Math, les containers, les algorithmes, les entrées/sorties et le multithreading.

Exemples de code et prérequis

Les exemples de code sont en téléchargement libre à l'url suivante : https://github.com/ChristophePichaud/Aide-Memoire-CPP-Moderne.

Pour compiler les exemples, il faut disposer de Visual Studio ou GCC sous MinGW64 ou WSL (Windows Subsystem for Linux) sous Windows. Sous Linux, un compilateur comme GCC ou Clang sera suffisant. Il s'agit de C++ ISO donc portable et multiplateforme.

À propos des compilateurs

Au moment de l'écriture de ce livre (novembre 2019), les versions des compilateurs étaient :

- Visual C++ 2019 v16.3.9 ;
- GCC 7.5 ;
- Clang & LLVM 9.0.

Le support C++17 est complet pour l'ensemble des compilateurs. Le support C++20 est partiel à l'heure actuelle mais le comité ISO C++ a remis les spécifications de C++20. La maturité des compilateurs comme GCC, Clang ou Visual C++ exploitent les derniers processeurs et toutes les architectures matérielles (ARM, x86, amd64, PPC, MIPS, Alpha, IA64).

Remerciements

Ce livre a été relu et corrigé par de nombreux amis et relations de travail. Parmi eux, merci plus particulièrement à Alain Zanchetta (Microsoft Corp.) qui est l'auteur de la préface mais surtout avec qui les chapitres 1 à 14 de la partie I, ont été relus en l'espace record de 3 semaines. Sachant que le temps d'Alain est précieux, il m'a confié que son temps de déjeuner quotidien sur le campus Microsoft était consacré à la relecture du livre sur sa Surface Pro 7. Ses remarques et commentaires ont permis de rendre le livre plus clair, plus juste et c'est un effort considérable que je salue avec tout le respect qu'il se doit. Encore une fois, merci !

Merci aussi à Éric Mittelette (Microsoft Corp.) et Jean-Christophe Godefroy (Altran) pour leurs instants de vérité et leur justesse.

Je tiens aussi à remercier Sy Brand (Microsoft Corp.), Stephan T. Lavavej (Microsoft Corp.) et Marian Luparu (Microsoft Corp.), membres de l'équipe Visual C++ pour leur patience et leurs réponses à mes questions sur le forum MVP.

Merci aussi à Éric Vernié (Microsoft France) qui m'a donné ma chance et permis de faire speaker à de nombreuses reprises sur les TechDays en France de 2011 à 2014 et qui a été mon mentor chez Microsoft en 2017. Éric a toujours été là pour défendre la place de C++ et prenait soin à chaque fois de me faire une place pour une session sur C++, que ce soit avec Alain, Loïc Joly ou en solo.

Merci à Raphael Mansuy (Caliatis), Daniel Begue (Orano), Fréderic Steczycki, Jean-Pierre Gervasoni (SyDev), Michel Foucault (freelance) et Sylvain Pontoreau.

Je tiens aussi à remercier mes anciens collègues de Sogeti que sont François Merand, Keelan Clech, Jean-Baptiste Bron et Cédric Georgeot. François tient une place toute particulière pour moi car quand il m'a recruté chez Sogeti en 2011 pour m'intégrer chez les *Rangers by Sogeti*, il voulait créer une cellule d'expertise sur les technologies Microsoft et voulait avoir des « experts communicants », c'est-à-dire des consultants qui font du blogging, des événements, des articles techniques, des livres et qui sont « *on the edge* », qui passent des certifications, qui deviennent MVP et influent dans l'écosystème Microsoft. C'est grâce à lui que j'ai donné mes premiers *talk* comme *speaker* et que je me suis mis à l'écriture d'articles techniques pour la revue *Programmez !* de François Tonic. Il m'a donné goût à l'écriture technique. François, tu as changé ma vie ! Ce livre est aussi une contribution dans mon activité de Rangers, une communauté technique qui existe toujours et qui est devenu indépendante[1] ; j'en suis le Community Manager.

Merci à Martine Thiphaine, ma Community Manager du programme Microsoft MVP[2] qui m'a annoncé que j'étais MVP Developer Technologies

1 www.netazurerangers.com.

2 *Microsoft Most Valuable Professional* (http://mvp.microsoft.com).

2019-2020 pour la deuxième année de suite. Merci, ça récompense une activité communautaire de partage autour de C++ et Windows. Ce livre est aussi une contribution pour mon profil MVP.

Mes meilleurs amis sont aussi à remercier car je ne les ai pas ménagés durant ces trois mois d'écriture : Philippe Rogie (Johnbry), Philippe Brand (freelance) et Emeric Blumberger (SocGen). Ils ont eu à subir mes sautes d'humeur fréquentes et mes coups de fils intempestifs en pleine journée…

Je tiens aussi à remercier le meilleur d'entre nous sur C++, à savoir mon ami Olivier Favre-Simon qui m'a fait découvrir Gentoo, Ubuntu, GCC, Boost et avec qui je suis impatient de refaire des soirées cigare & whisky.

Enfin, merci à Jean-Luc Blanc, mon éditeur chez Dunod qui a cru en ce projet. Il a tout de suite été convaincu de l'utilité d'avoir un ouvrage en français sur le C++ moderne. Merci aussi à Maxine Pouzet de chez Dunod pour la relecture finale. Maintenant que ce projet a pris forme, je réalise que c'était possible.

Enfin, je ne remercierais jamais assez mon père de m'avoir fait découvrir le monde des ordinateurs et des logiciels quand j'étais tout jeune : Commodore C64 (les jeux), Amiga (AMOS Basic, Aztec C, Lattice C/C++), PC 286 SX25 (Borland Turbo C/++ sous Dos), puis Windows 3.11 et Visual C++ 15.52c en 16 bits.

Christophe Pichaud (christophep@cpixxi.com),

Paris, France.

Vendredi 20 décembre 2019

À propos de l'auteur

Christophe Pichaud est un développeur et vit à Paris. Dans sa carrière, il a construit de grosses infrastructures bancaires, participé à l'ouverture de la première banque en ligne (Banque Populaire) et à la construction des services bancaires pour 2 500 agences Société Générale (MAIA, URTA). Il réalise aussi des migrations C++ et implémente des applications hybrides avec la stack Microsoft .NET. Ils travaille pour des clients comme Accenture, Avanade, Microsoft, Sogeti, Capgemini, le Palais de l'Élysée, SNCF, Total, Danone, CACIB et Bnp Paribas. Il possède les certifications MCSD et MCSD.NET. De plus, il participe aux événements Microsoft en tant que speaker (TechDays, DevDays) et MVP sur les stands Ask The Expert.

Christophe contribue régulièrement au magazine *Programmez* depuis 2011. Il est aussi le Community Manager des « .NET Azure Rangers », qui rassemblent 26 membres parmi lesquels 8 MVP et dont les activités sont l'animation de sessions techniques, l'écriture d'articles techniques et la promotion des technologies Microsoft ou Cloud.

Christophe travaille chez **Infeeny**[1], une filiale du groupe Econocom spécialisée dans les Technologies Microsoft. Quand il ne lit pas des livres ou développe du logiciel, Christophe passe son temps avec ses trois filles, Edith, Lisa et Audrey et avec ses parents.

1 https://infeeny.com/.

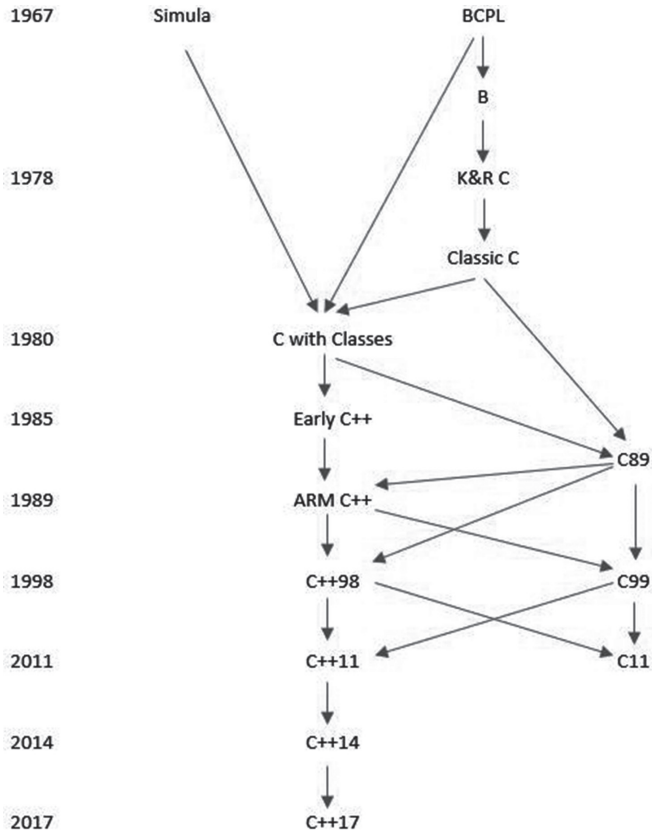
Introduction à C++

C++ est aujourd'hui un langage de programmation populaire. Créé en 1979, sa population était de 400 000 en 1991. Depuis, les estimations en comptaient environ 4,5 millions en 2018. La plus grosse évolution est apparue en 2005, lorsque les processeurs ont arrêté de croître exponentiellement en termes de performance : la performance des langages devenait alors importante. Cette évolution du nombre de développeurs s'est faite sans effort marketing ni support de communauté.

C++ est avant tout un langage industriel ; il est davantage utilisé dans l'industrie que dans les universités ou dans la recherche. Il a évolué chez Bells Labs et a été inspiré par les besoins de programmation système (*devices drivers*, réseau, systèmes embraqués). À partir de là, C++ s'est diffusé partout dans l'industrie : l'électronique, les infrastructures et les applications Web, les systèmes d'exploitation, la finance, le médical, l'automobile, l'aérospatiale, la physique, la biologie, l'énergie, le *machine learning* et l'intelligence artificielle, les jeux vidéo, l'animation graphique, la réalité virtuelle, etc. Il est utilisé partout où il faut exploiter l'architecture matérielle des processeurs de manière efficace pour réaliser des opérations complexes.

I. 1 Histoire de C++

Le C++, né en 1979, a été créé par Bjarne Stroustrup chez Bell AT&T. Depuis, il a subi plusieurs grandes étapes, comme C++98, C++11, C++14 et C++17 :



Comme on peut le constater sur le schéma ci-dessus, ISO C et ISO C++ ont un même parent : le C.

Le nouveau standard C++20 est attendu pour février 2020 et durant le premier semestre 2020, les éditeurs de compilateurs vont finaliser leurs implémentations. À la date d'écriture de cet ouvrage, le standard est finalisé mais les sociétés qui font les compilateurs n'ont pas encore terminé de l'implémenter. Cet ouvrage évoque les notions de C++20 qui sont en cours d'implémentation.

I. 2 Le C++ moderne

Le C++ moderne est apparu avec le langage ISO C++11. C'est un langage simple qui s'écrit comme du Java ou du C#. Avec la bibliothèque standard STL (*Standard Template Library*), le langage dispose de bon nombre de subtilités qui lui permettent d'être procédural, objet, fonctionnel ou méta-langage. Les standards C++11, 14, 17 et bientôt 20 nous apportent de nombreuses fonctionnalités que nous allons découvrir : gestion automatique de la mémoire *via* des pointeurs intelligents (Smart Pointers), déduction de type automatique à la déclaration *via* auto, etc.

I. 3 Les efforts du marketing et les langages « productifs »

À une époque, on disait que le C++ était *unsafe* et *unsecure*. Le but était de vendre des langages comme Java ou C#, dits « productifs », de plus haut niveau et basés sur une runtime et un Framework et qui souvent cherchaient, pour se faire un nom, à décrédibiliser C++. Par défaut, C++ est rapide, il gère les exceptions et possède la faculté de réaliser des applications orientées objet très sophistiquées. Il est très difficile de lui trouver des défauts. Malgré tous ces efforts de marketing, C++ a toujours eu du succès. Les langages dits productifs sont des dérivés de C++ avec une syntaxe très proche voire équivalente et n'ont piochés dans C++ que ce qu'il y a de simple et efficace. Préférez l'original à la copie.

I. 4 Le marché des logiciels grand public

Pour expliquer pourquoi le langage C++ est roi, il suffit d'observer les logiciels que nous utilisons au quotidien et se demander dans quel langage ils sont écrits : Windows , Word, Excel, PowerPoint, Outlook, Chrome, VLC, Acrobat PDF Reader, Photoshop, Java JVM, Microsoft .NET CLR,

Apache, IIS, etc. Tous ces systèmes d'exploitations et logiciels sont faits en C/C++. Le point essentiel ici est que le noyau d'un système d'exploitation est écrit en C et non en C++.

Dans les index de popularité des langages comme TIOBE (https://www.tiobe.com/tiobe-index/), le C représente 12,5 % et le C++ 7,5 %. Java représente 15,9 % et C# 2,8 %. Le couple C/C++ totalise donc 20 % et ce depuis des décennies.

C++ est à la fois bas niveau – avec C et l'assembleur – et haut potentiel d'abstraction. Le code C++ est traduit en assembleur et les optimisateurs de code C++ exploitent les extensions des processeurs avec des instructions étendues et parallèles pour un code de qualité supérieur qui tire parti des processeurs x86, x64 et ARM pour ne citer qu'eux. Il existe différents vendeurs de compilateurs : Intel, Microsoft, IBM et la communauté GNU avec son fameux GCC et n'oublions pas le dernier, Clang de la famille LLVM.

Le C++ est normalisé à travers le comité ISO présidé par Herb Sutter. Il y a eu plusieurs standards comme C++03, C++11, C++14, C++17, etc. À partir de C++11 on assiste la renaissance de C++ : c'est l'avènement du C++ moderne que nous allons détailler dans le présent ouvrage.

1

Le langage

Dans cette partie, nous allons étudier le langage C++ avec ses fondamentaux :

- les variables et les types ;
- les déclarations ;
- les fonctions ;
- les classes ;
- les opérations de copie et déplacement ;
- les opérations de surcharge ;
- la programmation orientée objet ;
- les templates.

Le chapitre 3 est un avant-goût de tous les chapitres sur le langage. Il présente l'essentiel pour mettre le lecteur en situation de coder.

1 Les fondamentaux

Le C++ est un langage compilé qui génère de l'assembleur et produit des fichiers objets qui sont ensuite utilisés par l'éditeur de liens pour construire un exécutable directement exploitable par le processeur. Il n'y a pas de génération d'un assembleur intermédiaire nécessitant un interpréteur ou un mécanisme de compilation « Just In Time ». Sous Windows, on produit des EXE ou des DLL, sous Linux on génère des exécutables ou des fichiers partagés so.

Le standard C++ possède deux volets :

- le langage ;
- la bibliothèque standard STL.

La STL est construite en C++ et est livrée avec chaque compilateur. Il existe une version open source de la STL comme Boost qui est très populaire (www.boost.org). Boost fournit d'autres choses en plus de la STL.

C++ est un langage typé statique. Tous les éléments doivent être connus à la compilation.

Dans ce chapitre, nous ferons un tour rapide des grands éléments qui constituent le socle de C++, à savoir les fonctions, les types, les pointeurs, les références et les classes.

1.1 Mon premier programme

Le point d'entrée d'un programme est la fonction main() :

```
#include <iostream>


int main()
{
    std::cout << "Hello World!\n";
    return 0;
}
```

La première ligne du *main*, qui est le point d'entrée, demande d'inclure la librairie d'entrée-sortie <iostream>. Elle montre l'utilisation de l'objet cout présent dans l'espace de nom (namespace) std. Plus précisément, l'opérateur << est utilisé pour passer la chaîne « Bonjour » en paramètre. Il est possible de rendre le code plus lisible en demandant que l'espace de nom soit utilisé sans le mentionner *via* la directive *using namespace std*.

```
using namespace std;


int main()
{
    cout << "Hello World" << endl;
    return 0;
}
```

endl signifie le passage à la ligne. On remarque qu'il est possible d'utiliser l'opérateur << plusieurs fois.

1.2 Ma première fonction

Dans le code, il est possible de faire différentes fonctions, comme en C, pour séparer les traitements :

```
void SayHello()
{
    cout << "Hello" << endl;
}

int main()
{
    SayHello();
    return 0;
}
```

1.3 Les fonctions

Une fonction qui ne retourne rien est préfixée de « void ». Une fonction ne peut pas être appelée si elle n'a pas été déclarée. Dans l'exemple ci-dessous, le prototype de la fonction *SayHello2()* est positionné avant la fonction *main()*. Le prototype d'une fonction, correspond à la ligne qui contient le type de retour, le nom de la fonction et les arguments suivis d'un point-virgule « ; ».

La déclaration se fait comme suit :

```
void SayHello2();

int main()
{
    SayHello2();
    return 0;
}

void SayHello2()
{
    cout << "Hello2 !" << endl;
}
```

Lorsque la fonction possède des arguments, il n'est pas nécessaire de les nommer dans le prototype :

```
void RepeatHello(int);  // Repeat N times


int main()
{
    RepeatHello(5);
    return 0;
}


void RepeatHello(int n)
{
    for (int i = 0; i < n; ++i)
    {
        cout << "Hello2 !" << endl;
    }
}
```

Deux fonctions peuvent avoir le même nom à condition qu'elles se différencient par le type ou le nombre de leurs paramètres ; le type retourné n'est en revanche pas considéré comme différenciateur. C'est le compilateur qui fera l'appel à la bonne fonction.

```
void DisplayPrice(int price);
void DisplayPrice(double price);



int main()
{
    DisplayPrice(10);
    DisplayPrice(25.401);
    return 0;
```