Christophe **Moor**

Mémento du langage C

Norme ANSI/ISO

2e version

Nouvelles fonctionnalités C99 + C11

BOOKELIS

Historique des versions

Version 1.1 (août 2018)

— Ajout des chapitres 6, 7, 8, et 9
— Finalisation des extensions C99
— Ajout des extensions C11
— Nouvelle mise en page

Version 1.0 (septembre 2016)

Version initiale

Contacter l'auteur

moorchr@gmail.com
chrismoor

Publication Bookelis, coll. Copernic
38 Parc du Golf
FR-13290 Aix-en-Provence

ISBN : 979-10-227-8123-7

Les illustrations présentes en couverture, libres de droits, sont issues de la médiathèque *Wikimedia Commons*, ainsi que du moteur de recherche *Pexels*.

Mise en page avec LaTeX 2_ε.

Version actuelle : 1.1 rev. 2

Préambule

* * *

Pourquoi ce mémento ?

Le but de cet opuscule est de rassembler tous les points clés essentiels à une bonne compréhension du langage C de manière condensée, et d'y exposer des éléments de syntaxe parmi les plus utiles au quotidien, jusqu'aux notions avancées. De plus, les points délicats et les ambiguïtés du langage les plus couramment rencontrées sont mis en exergue, de manière à éviter les erreurs qui pourraient être induites par la pratique d'un autre langage de programmation. Dans la mesure du possible, nous mentionnons aussi les risques *pratiques* – effets constatés avec les compilateurs modernes – encourus en cas de non-respect de la norme.

Pour qui ?

Cet abrégé s'adresse notamment aux programmeurs occasionnels (en C) qui souhaitent trouver une réponse rapide aux problèmes techniques rencontrés lors du développement d'applications, ou aux personnes désireuses de rafraichir rapidement leurs connaissances, quitte à aller consulter un ouvrage de référence afin d'obtenir davantage de détails au sujet de certaines notions en particulier.

Ce mémento ne prétend pas à l'exhaustivité !

L'ouvrage que vous vous apprêtez à lire n'est ni une documentation technique, ni un cours complet sur le langage C, ni encore moins un cours de programmation. De solides notions de programmation, que ce soit en C ou non, sont requises avant d'aborder le contenu qui suit. Par ailleurs, tous les commentaires constructifs seront bienvenus et appréciés, qu'ils soient relatifs à des idées d'amélioration, ou à des erreurs relevées.

TABLE DES MATIÈRES

INTRODUCTION

Bref historique

Le langage C fut initialement inventé pour écrire le système d'exploitation UNIX au sein des laboratoires Bell, au début des années 1970. Dennis Ritchie a fait évoluer le langage B – développé par Ken Thompson quelques années auparavant, qui s'est lui-même inspiré de BCPL – en y ajoutant notamment la notion de type. Par la suite, Brian Kernighan a aidé à populariser le langage et a en faire la version connue aujourd'hui sous le nom de « C K&R ». Cette dernière fut normalisée en 1989 par l'ANSI après quelques modifications mineures, ainsi que la définition d'une bibliothèque (inexistante auparavant), et constitue le C moderne tel qu'il est encore utilisé aujourd'hui.

Principales caractéristiques du langage

Le langage C est efficace, portable et proche de la machine. On parle de langage « bas niveau », car il manipule les mêmes types d'objets que la machine – les caractères, les nombres, et les adresses – et qu'ils sont combinables à l'aide d'opérateurs arithmétiques et logiques que la machine fournit réellement. En particulier, le C ne fournit aucune opération sur les types composés, tels les chaines de caractères ou les tableaux. Il est compilé de manière à ce que chaque instruction exécutable soit traduite par un nombre relativement prévisible d'instructions-machine, ce qui a une incidence sur l'occupation en mémoire et le temps de calcul. Par ailleurs, le langage C à proprement dit ne permet pas de réaliser des entrées-sorties ; à titre d'exemple, il n'existe aucune méthode ni aucun opérateur intrinsèque d'accès aux fichiers, mais ce genre d'opération est réalisable par l'intermédiaire de fonctions proposées par les diverses implémentations de la bibliothèque standard [1] (glibc, MSVCR, ...). Comme la priorité est donnée à la légèreté et à la portabilité des applications, aucun mécanisme de multiprogrammation ou de synchronisation ente tâches n'est proposé de façon normalisée, tout autant que la description d'interfaces graphiques ou la gestion des exceptions, entre autres.

Malgré le peu de fonctionnalités, le C est indépendant de toute architecture matérielle particulière, la taille de sa bibliothèque très petite, et son apprentissage relativement rapide. Du fait de la faible quantité de concepts inhérents au langage, les compilateurs restent très simples, ce qui leur permet d'être rapides et facilement portables d'une architecture à une autre. D'ailleurs, un compilateur C est très souvent nativement livré avec les systèmes d'exploitation de la famille UNIX.

1. Bibliothèque logicielle du C livrée avec tout compilateur respectant la norme ISO, et parfois avec le système d'exploitation. Elle contient une collection élémentaire de fonctions mathématiques, de manipulation de chaines de caractères, de conversion de types, d'entrées/sorties, d'allocation mémoire, etc.

La puissance et la flexibilité de ce langage font qu'il est utilisé par beaucoup de compilateurs et interpréteurs à destination d'autres langages de programmation (p. ex. Fortran, Perl, Python, PHP).

Quelques avantages du C

— Langage très répandu et largement connu ;
— bâti sur un standard ouvert ;
— gestion de la mémoire/maximisation des performances ;
— aucun support à l'exécution n'est requis (p. ex. machine virtuelle) ;
— code portable et compact ;
— etc.

Quelques inconvénients du C

— Difficulté à détecter les erreurs si l'on ne fait pas attention (ou moyennant des logiciels tiers), et aucune vérification à l'exécution (sauf en C11) ;
— modularité sommaire (p. ex. absence d'espace de noms) ;
— gestion rudimentaire des exceptions ;
— généricité quasi absente (un peu plus étendue en C11) ;
— aucune gestion de la mémoire incorporée ;
— etc.

En outre, le langage C est très permissif, et le code source peut très rapidement devenir illisible. On pourrait aussi lui reprocher le manque de productivité que ses avantages et inconvénients induisent face à d'autres langages plus modernes.

Remarques au sujet des normes ISO 9899

Le comité de normalisation a établi des lignes directrices pour l'écriture des normes ISO relatives au langage C. Voici un extrait déterminant ce qui caractérise **l'esprit du langage C** :

— fais confiance au programmeur ;
— n'empêche pas le programmeur de faire ce dont il a besoin ;
— garde le langage simple et concis ;
— ne propose qu'une seule option pour effectuer une opération donnée ;
— fais-le rapidement, même si ce n'est pas complètement portable.

Attention à certaines fonctionnalités introduites en C99 – principalement les tableaux à dimensions variables et les types pour les nombres complexes – dont le support n'est plus obligatoire en C11. De plus, la prise en charge de quelques grandes nouveautés en C11 est optionnelle ; on peut citer le *multithreading*, les objets atomiques, ou encore les interfaces de vérification des limites (*bounds checking*). La dernière mouture de la norme (C17/C18), publiée en 2018, ne fait que corriger les défauts de C11 (y. c. les modifications de 2012) et clarifie certains aspects [2] ; aucune nouvelle fonctionnalité n'est ajoutée.

Contenu de cet abrégé

Une très large majorité des règles normalisées sont présentées, plus ou moins en détail selon le degré d'utilité. Par contre, cet ouvrage n'a pas vocation à devenir une référence de la bibliothèque standard [3], et donc seulement quelques fonctions typiques – surtout celles dont la compréhension correcte pourrait être délicate, ou celles sujettes à un usage intensif – sont décrites.

2. Liste complète : `http://www.open-std.org/jtc1/sc22/wg14/www/abq/compendium.htm`
3. À cet effet, pensez à utiliser la commande "`man` *fonction*", disponible sur les systèmes UNIX.

Les nouveautés apportées par les versions plus récentes de la norme sont repérables par les symboles C 99 et C 11. Toutes les nouveautés inhérentes au langage sont abordées, ce qui n'inclut pas certaines nouveautés de la bibliothèque (fonctions, ou paramètres/concepts exclusivement liés à ces fonctions), car ce n'est pas le but de ce mémento. En C11, on n'aborde pas non plus les concepts de *multithreading*, ni l'interface de vérification des limites ou celle d'analysabilité.

Conventions de notation

Pour une plus grande facilité de lecture, le contenu qui suit est structuré en **points-clés** dont la signification est la suivante :

- ◆ Règle dictée par la norme, ou présentation d'une fonction
  - ▪ Niveau hiérarchique 2
    - ◇ Niveau hiérarchique 3
- ↻ Valeur de retour d'une fonction
- → But / finalité
- ⇆ Comparaison avec un autre point, fonction, langage, etc.
- ⇒ Conséquence
- ▷ Précision utile / conseil / exemple

Certains paragraphes apportent des **explications supplémentaires**, en essayant d'anticiper les questions/erreurs courantes, et en conseillant quelques bonnes pratiques de programmation :

Informations complémentaires non indispensables

Conseils pour éviter les pièges, mises en garde

À propos des extraits de code

Dans tout ce qui suit, les commentaires seront dénotés avec la syntaxe de fin de ligne (`//`) pour éviter de surcharger les exemples, bien que la norme C90 n'autorise uniquement les commentaires sous forme de blocs `/* ... */`. Néanmoins, les deux syntaxes sont acceptées par la grande majorité des compilateurs modernes.

Dans le même but, on a volontairement éludé toutes les éventuelles directives d'inclusion des fichiers en-tête de la bibliothèque standard (*stdio.h*, *stdlib.h*, *stdarg.h*, *math.h*, *string.h*, *time.h*, *ctype.h*, *limits.h*, *float.h*), sauf s'il s'agit d'un cas particulier. Aussi, les en-têtes de fonctions (typiquement *main*) sont ôtés dans les exemples les plus succincts.

Version de la norme Toute implémentation conforme au standard C doit définir la macro constante `__STDC__` avec la valeur 1. Avec la norme C99, la macro `__STDC_VERSION__` vaut $199901L$ (constante entière), tandis qu'elle vaut $201112L$ en C11, et $201710L$ en C18. Cette macro n'est pas définie en C89/C90 pur, mais fut introduite avec l'amendement de 1995 ($199409L$).