

[Le Langage C](#)

[Introduction](#)

[1 - Premier programme en C](#)

[2 - Un exemple de programme plus évolué](#)

[3 - Variables-constantes-affectation](#)

[4 - Opérateurs - Conversions](#)

[4.1. Quelles sont les priorités quand on mélange des opérateurs ?](#)

[4.2. Les opérateurs arithmétiques : + - \\* / % \(modulo\)](#)

[4.3. L'opérateur d'affectation = \(« reçoit »\)](#)

[4.4. Les conversions de type : implicites et explicites \(cast\)](#)

[4.4.1 Les conversions implicites](#)

[4.4.2 Les conversions explicites : l'opérateur de cast](#)

[4.5. Les opérateurs relationnels : inférieur, supérieur, égal, différent...](#)

[4.6. Les opérateurs logiques : ET OU NON](#)

[4.7. Opérateurs de manipulation de bits – masques \(ET bit à bit, décalage...\)](#)

[4.8. L'opérateur d'adresse &](#)

[4.9. Les opérateurs d'incrément et de décrémentation ++ --](#)

[4.10. L'opérateur sizeof \(taille en octets\)](#)

[5 - Les structures de contrôle](#)

[6 - Les entrées/sorties](#)

[7 - Utilisation de fonctions](#)

[8 - La bibliothèque de fonctions \(sinus, exp, valeur absolue...\)](#)

[9 - Définition de fonction](#)

[10 - La compilation séparée \(multi-fichiers\)](#)

[11 - Les tableaux](#)

[12 - Les chaînes de caractères](#)

[13 - Les pointeurs](#)

[14 - Pointeurs et tableaux à une dimension](#)

[15 - Pointeurs et chaînes de caractères](#)

[16 - Les structures](#)

[17 - Les fichiers](#)

[18 - Les simplifications d'écriture](#)

[19 - Les classes d'allocation mémoire](#)

[20 - Etes-vous un « bon » programmeur ?](#)

[Annexes](#)

[Moteur de recherche](#)

4.4. Les conversions de type : implicites et explicites (cast)



Les conversions de type : implicites et explicites (cast)

Le Langage C est très (trop) tolérant en ce qui concerne les mélanges de types dans une expression. C'est au programmeur de vérifier que les **conversions implicites** réalisées par le compilateur ont le sens désiré... Cet allègre mélange des types est un des plus gros pièges pour le débutant, à qui un langage fortement typé convient mieux. En l'absence de cadre rigoureux, il est facile de réaliser sans s'en douter des opérations hasardeuses... et fausses.

Le programmeur confirmé, lui, utilisera comme un atout ce mélange possible.