Langage C

4.4.2 Les conversions explicites : l'opérateur de cast

Icône de l'outil pédagogique Les conversions explicites : l'opérateur de cast

La conversion implicite effectuée par le compilateur, ou le type utilisé pour un calcul, ne convient pas toujours au programmeur : celui-ci peut forcer la conversion d'une expression quelconque **dans le type de son choix** grâce à l'opérateur unaire de conversion appelé cast. C'est par exemple très utile pour obliger un calcul à se faire en réel bien que ses opérandes soient des entiers (piège de la division entière…).
Sa syntaxe d'un cast est la suivante : **(type) expression**

Exemple 10, Conversions explicites par « cast » (télécharger le c des 4 écritures)

Le cast s'applique à la valeur 1000 qui est converti en *long* → cela permet au reste du calcul de se faire en *long* et ainsi d'**éviter un dépassement** lors de la multiplication. Le résultat de la division finale sera converti si nécessaire lors de l'affectation dans le type de la Leftvalue entier.

Nous avons ici un cas classique de **règle de trois** pour une mise à l'échelle entre 0 et 100 d'un entier compris à l'origine entre 0 et NMAX. Le cast de l'opérande n en double permet de faire t**out le calcul en réel**. Le résultat sera converti en entier lors de l'affectation finale (voir exemple suivant).

Ce cast permet de convertir la variable réelle *var_double* en entier en « tronquant" la partie décimale.
Attention : ce n'est pas un arrondi à la valeur la plus proche : 3.99 sera converti en 3 (ce qui correspond bien à la définition de la partie entière en mathématiques).
Subtilité pour les nombres négatifs : -3.99 sera converti en -3 (alors que les mathématiques définissent la partie entière comme -4 dans ce cas).

Ce cast permet de réaliser un arrondi à la valeur la plus proche : 3.99 sera arrondi en 4 et 3.49 sera arrondi en 3. Attention : cela ne marche que pour les valeurs positives.

Terminons avec les pièges de la division entière.

Exemple 11, Division entière et conversion explicite par « cast » (télécharger le c des 2 écritures)

Soient *n* et *p* deux variables **entières** valant 10 et 3. On cherche à effectuer leur division **réelle** et à stocker le résultat dans une variable réelle res_reelle.

Solution 1 (fausse) :

l'expression entière n/p est ici convertie en *double* **après** division et res_reelle vaut 3.0. En raison des parenthèses, l'opérateur force la conversion du **résultat** de l'expression et non celle des valeurs qui la composent. La division reste une divison entière et notre but n'est pas atteint.
Notons que le cast explicite est superflu, car le compilateur réalise de toute façon cette conversion.

Solution 2 (juste) :

le cast s'applique ici sur le premier opérande *n* et permet à la division de s'effectuer en *double* alors qu'elle s'effectuait précédemment en entier. Cette fois, nous n'avons plus une division entière, mais une division réelle. Au final, *res_reelle* vaut 3.3333.

☒ Un cast *(short int)* ou *(long int)* effectué sur un réel revient à prendre la partie entière de la valeur absolue, puis à « remettre le signe » (sous réserve que la valeur initiale reste dans les limites du nouveau type). C'est parfois bien utile.

☒ Un cast *(char)* sur un entier revient à prendre sa valeur modulo 256.

« Précédent | Suivant »