C - Les nombres entiers (`int`)

Les **types entiers** en C permettent de représenter des **nombres sans virgule**, **positifs ou négatifs**. Le C offre des versions **signées** et **non signées** (`unsigned`) pour contrôler la présence ou non de nombres négatifs. Bien comprendre ces différences est essentiel pour écrire des programmes efficaces et fiables.

5TTR 6TTR 4GMS Découverte

Les entiers (`int` , `unsigned int`)

Les entiers permettent de stocker des nombres entiers (sans virgule), positifs ou négatifs.

Les **int** stockent des valeurs positives et/ou négatives (ils utilisent un bit de signe).

Les **unsigned int** stockent des valeurs positives uniquement

La taille d'un `int` en C dépend de l'architecture de la machine, mais il est généralement de **4 octets sur les systèmes modernes 32 bits et 64 bits**. La taille peut être de 2 octets sur des systèmes 16 bits plus anciens ou embarqués.

Type	Taille (16 bits)	Valeur minimale (16 bits)	Valeur maximale (16 bits)	Taille (32/64 bits)	Valeur minimale (32/64 bits)	Valeur maxim
`int`	2 octets (16 bits)	−32 768	+32 767	4 octets (32 bits)	−2 147 483 648	+2 147 483 64
`unsigned int`	2 octets	0	65 535	4 octets	0	4 294 967 295

À retenir

- La taille d'un type dépend du **compilateur** et de la **plateforme**.
- Un `unsigned` ne peut pas représenter de nombres négatifs.
- Utilise `sizeof(type)` dans ton programme pour vérifier la taille réelle :

```c
printf("int = %zu octets\n", sizeof(int));
```

Exemples

```c
int temperature = - 10;
unsigned int vitesseMoteur = 100; // Vitesse initiale du moteur
```