Bornes des entiers en Rust
En Rust (comme dans beaucoup de langages), les types numériques ont des bornes fixées par leur taille en bits. Voici un récapitulatif des bornes pour les types signés (*i*) et non signés (*u*).

Type	Bits (n)	Plage de valeurs
i8	8	–128 à 127
u8	8	0 à 255
i16	16	–32 768 à 32 767
u16	16	0 à 65 535
i32	32	–2 147 483 648 à 2 147 483 647
u32	32	0 à 4 294 967 295
i64	64	–9 223 372 036 854 775 808 à 9 223 372 036 854 775 807
u64	64	0 à 18 446 744 073 709 551 615
i128	128	–2^127 à 2^127 – 1
u128	128	0 à 2^128 – 1
isize	dépend de l'architecture	–2^(n–1) à 2^(n–1)–1
usize	dépend de l'architecture	0 à 2^n – 1

Formules générales :
- Pour un entier signé (*iN*) : la plage est de -2^{n-1} à $2^{n-1} - 1$.
- Pour un entier non signé (*uN*) : la plage est de 0 à $2^n - 1$.

Exemple :
- i8 (8 bits) → –2^7 = –128 jusqu'à 2^7 – 1 = 127.
- u8 (8 bits) → 0 jusqu'à 2^8 – 1 = 255.

■■ Dépassement d'entier (Integer Overflow)
Quand on dépasse les bornes d'un type entier, Rust adopte un comportement précis : **En mode debug** → le programme panique (arrêt avec erreur). **En mode release** → encapsulage automatique (*wrapping*), la valeur repart de zéro ou passe au minimum.

Opération	Résultat attendu	Résultat en Rust
255u8 + 1 (debug)	256 (impossible)	Panique ■■
255u8 + 1 (release)	256 (impossible)	0 (wrapping)
255u8.saturating_add(1)	reste bloqué au max	255
255u8.wrapping_add(1)	retour circulaire	0
255u8.overflowing_add(1)	retour + drapeau	(0, true)
255u8.checked_add(1)	résultat Option	None

Rust propose plusieurs méthodes pour contrôler explicitement le comportement lors d'un dépassement : **.wrapping_*()** → encapsulage circulaire. **.saturating_*()** → reste au maximum/minimum. **.overflowing_*()** → renvoie la valeur + un drapeau booléen d'overflow. **.checked_*()** → renvoie un *Option* (None si dépassement).