

# `stdlib` | Managing Integer Overflow

# Overflow

- ◆ Primitive integers can overflow when they reach their limits
  - In *debug* mode: panic
  - In *release* mode: wrap
- ◆ Wrapping may or may not be desired
- ◆ Functions exist to handle these situations
  - Communicate intent
  - Reduce bugs

# ■ About Functions

- ◆ Defined on integer types
  - 8, 16, 32, 64, 128 bit signed & unsigned
- ◆ Typical operations such as addition, division, multiplication each have functions to handle overflow
  - Each function has different behavior when overflow occurs

# Overflow Functions

- ◆ *checked\_\** **Option<i32>**
  - Returns an option
- ◆ *overflowing\_\** (**i32, bool**)
  - Indicates whether overflow occurred
- ◆ *saturating\_\**
  - Limits the value to the min or max of the type
- ◆ *wrapping\_\**
  - Wraps on overflow (default)

## ■ `checked_*`

```
// None
```

```
let n: Option<u32> = 0u32.checked_sub(1);
```

```
// None
```

```
let n: Option<u32> = u32::MAX.checked_add(1);
```

```
// Some(10)
```

```
let n: Option<u32> = 9_u32.checked_add(1);
```

## ■ *overflowing\_\**

```
// (4294967295, true)
```

```
let n: (u32, bool) = 0u32.overflowing_sub(1);
```

```
// (6, false)
```

```
let n: (u32, bool) = 5u32.overflowing_add(1);
```

## ■ *saturating\_\**

```
// 0
```

```
let n: u32 = 0_u32.saturating_sub(9001);
```

```
// 4294967295
```

```
let n: u32 = u32::MAX.saturating_add(u32::MAX);
```

## ■ *wrapping\_\**

```
// 4294967295
```

```
let n: u32 = 1_u32.wrapping_sub(2);
```

```
// 0
```

```
let n: u32 = u32::MAX.wrapping_add(1);
```

# ■ Recap

- ◆ Arithmetic overflow will panic in debug builds & overflow in release builds
- ◆ Overflow functions exist to handle these situations in different ways
- ◆ When performing arithmetic on numbers at the extremes, prefer using an overflow function to reduce bugs