Please place this text in a box after the “Integer Types” section ends and before the “Floating-Point Types” section begins on page 38.

Integer Overflow

Let’s say you have a variable of type u8 that can hold values between 0 and 255. If you try to change the variable to a value outside of that range, such as 256, integer overflow will occur. Rust has some interesting rules involving this behavior. When you’re compiling in debug mode, Rust includes checks for integer overflow that causes your program to panic at runtime if this behavior occurs. Rust uses the term panicking when a program exits with an error; we’ll discuss panics in more depth in the section “Unrecoverable Errors with panic!” in Chapter 9 on page XX.

prod: check xref

When you’re compiling in release mode with the --release flag, Rust does not include checks for integer overflow that cause panics. Instead, if overflow occurs, Rust performs two’s complement wrapping. In short, values greater than the maximum value the type can hold “wrap around” to the minimum of the values the type can hold. In the case of a u8, 256 becomes 0, 257 becomes 1, and so on. The program won’t panic, but the variable will have a value that probably isn’t what you were expecting it to have. Relying on integer overflow’s wrapping behavior is considered an error. If you want to wrap explicitly, you can use the standard library type Wrapping.

So with the wrap around, the value is just entirely changed, presumably resulting in unexpected behavior, is that right? I wasn’t clear on the consequence of unintentional wrapping