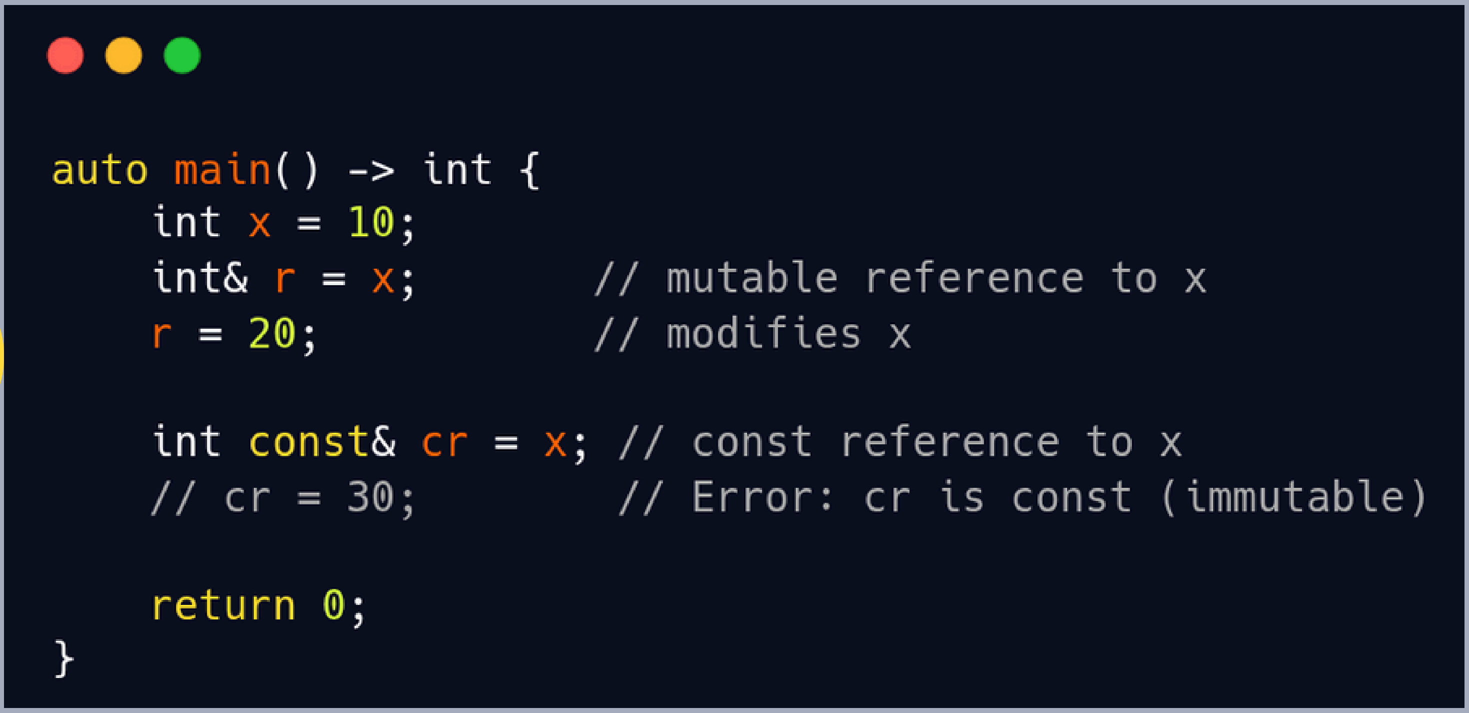
C++ to Rust

Follow this series for more bite-sized comparisons between C++ and Rust!

References & Mutability

C++

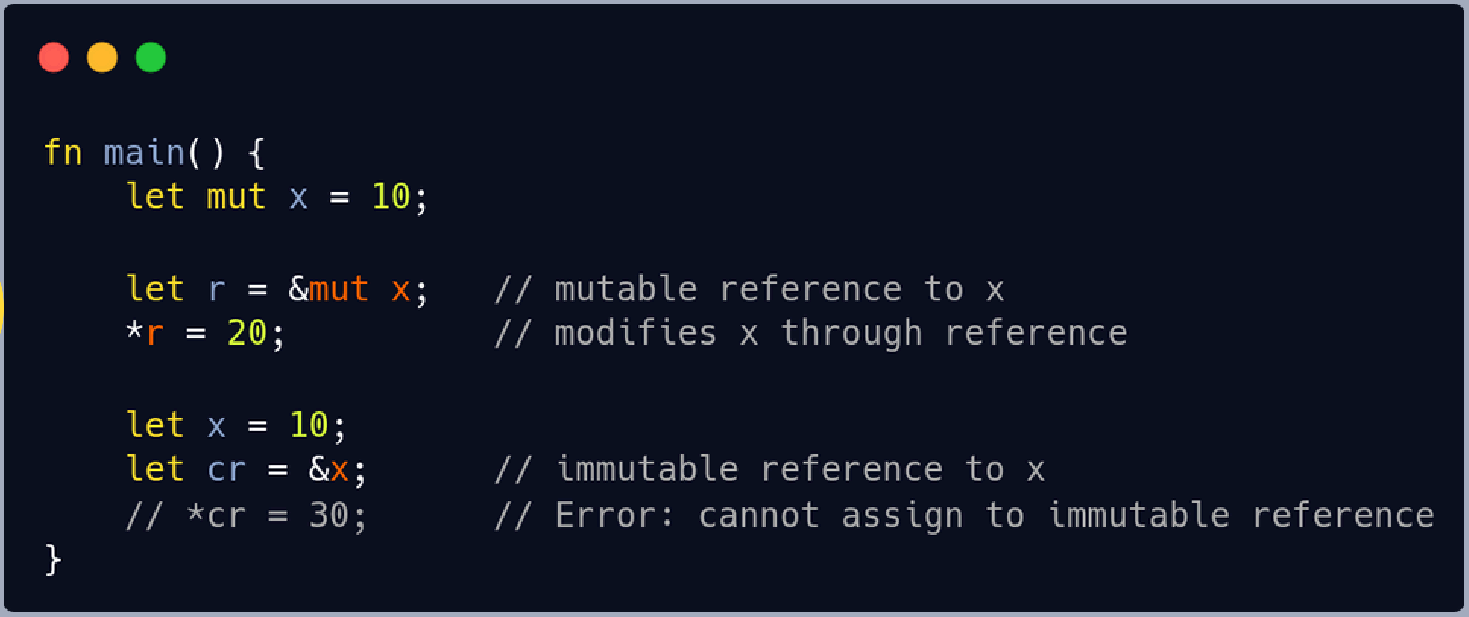


```cpp
auto main() -> int {
    int x = 10;
    int& r = x;        // mutable reference to x
    r = 20;            // modifies x

    int const& cr = x; // const reference to x
    // cr = 30;         // Error: cr is const (immutable)

    return 0;
}
```

Rust



```rust
fn main() {
    let mut x = 10;

    let r = &mut x;    // mutable reference to x
    *r = 20;           // modifies x through reference

    let x = 10;
    let cr = &x;       // immutable reference to x
    // *cr = 30;       // Error: cannot assign to immutable reference
}
```

C++ to Rust

Follow this series for more bite-sized comparisons between C++ and Rust!

✅ **What to notice:**

- 📌 In both C++ and Rust, references allow you to work with values without copying.
- 🔐 Rust enforces exclusive mutability: you can't have multiple references if one is mutable—***this avoids data races at compile time.***
- 📎 C++ references can be const or not, but the compiler doesn't enforce the same level of aliasing safety as Rust.
- 💥 Rust uses *r to dereference and assign, like C++ pointers, but with safety guarantees built into the borrow checker.

💬 ***Which one do you think makes references safer to use?***