A mutable reference was used in a constant.

Erroneous code example:

```
#![feature(const_mut_refs)]

fn main() {
    const OH_NO: &'static mut usize = &mut 1; // error!
}
```

Mutable references (`&mut`) can only be used in constant functions, not statics or constants. This limitation exists to prevent the creation of constants that have a mutable reference in their final value. If you had a constant of `&mut i32` type, you could modify the value through that reference, making the constant essentially mutable.

While there could be a more fine-grained scheme in the future that allows mutable references if they are not "leaked" to the final value, a more conservative approach was chosen for now. `const fn` do not have this problem, as the borrow checker will prevent the `const fn` from returning new mutable references.

Remember: you cannot use a function call inside a constant or static. However, you can totally use it in constant functions:

```
#![feature(const_mut_refs)]

const fn foo(x: usize) -> usize {
    let mut y = 1;
    let z = &mut y;
    *z += x;
    y
}

fn main() {
    const FOO: usize = foo(10); // ok!
}
```