Note: this error code is no longer emitted by the compiler.

You used a function or type which doesn't fit the requirements for where it was used. Erroneous code examples:

```
#![feature(intrinsics)]

extern "rust-intrinsic" {
    fn size_of<T>(); // error: intrinsic has wrong type
}

// or:

fn main() -> i32 { 0 }
// error: main function expects type: `fn() {main}`: expected (), found i32

// or:

let x = 1u8;
match x {
    0u8..=3i8 => (),
    // error: mismatched types in range: expected u8, found i8
    _ => ()
}

// or:

use std::rc::Rc;
struct Foo;

impl Foo {
    fn x(self: Rc<Foo>) {}
    // error: mismatched self type: expected `Foo`: expected struct
    //        `Foo`, found struct `alloc::rc::Rc`
}
```

For the first code example, please check the function definition. Example:

```
#![feature(intrinsics)]

extern "rust-intrinsic" {
    fn size_of<T>() -> usize; // ok!
}
```

The second case example is a bit particular: the main function must always have this definition:

```
fn main();
```

They never take parameters and never return types.

For the third example, when you match, all patterns must have the same type as the type you're matching on. Example:

```
let x = 1u8;

match x {
    0u8..=3u8 => (), // ok!
    _ => ()
}
```

And finally, for the last example, only `Box<Self>` , `&Self` , `Self` , or `&mut Self` work as explicit self parameters. Example:

```
struct Foo;

impl Foo {
    fn x(self: Box<Foo>) {} // ok!
}
```