Note: this error code is no longer emitted by the compiler. The type-checker needed to know the type of an expression, but that type had not yet been inferred.

Erroneous code example:

```
let mut x = vec![];
match x.pop() {
    Some(v) => {
        // Here, the type of `v` is not (yet) known, so we
        // cannot resolve this method call:
        v.to_uppercase(); // error: the type of this value must be known in
                          //        this context
    }
    None => {}
}
```

Type inference typically proceeds from the top of the function to the bottom, figuring out types as it goes. In some cases – notably method calls and overloadable operators like `*` – the type checker may not have enough information *yet* to make progress. This can be true even if the rest of the function provides enough context (because the type-checker hasn't looked that far ahead yet). In this case, type annotations can be used to help it along.

To fix this error, just specify the type of the variable. Example:

```
let mut x: Vec<String> = vec![]; // We precise the type of the vec elements.
match x.pop() {
    Some(v) => {
        v.to_uppercase(); // Since rustc now knows the type of the vec elements,
                          // we can use `v`'s methods.
    }
    None => {}
}
```