# What does it mean when the compiler says that it can't convert something to itself?

devblogs.microsoft.com/oldnewthing/20231227-00

December 27, 2023

Raymond Chen

A customer encountered a very strange error message from the Visual C++ compiler:

```
oops.cpp(7): Error C2664: 'void something(blah)' cannot convert argument 1 from
'blah' to 'blah'
```

Why is the compiler complaining that it cannot convert a `blah` to a `blah`? How is it not possible to convert something to itself?

The answer is given in the next line of the error message:

```
oops.cpp(7): note: use of undefined type 'blah'
```

Here's a sample program that demonstrates the problem.

```
struct blah;

void something(blah);

void test(blah& b)
{
    something(b); // error here
}
```

The problem is that the `test` function is passing a `blah` object by value to the `something` function. This requires the compiler to convert the thing you passed (a `blah` object) to the thing the function accepts (a `blah` object), which would normally be accomplished by using the copy constructor. But the compiler can't find a copy constructor for `blah`, so it complains.

Now, the reason it can't find a copy constructor is that the type `blah` has never been defined. All that exists is a forward reference. That's what the second error message is trying to tell you: "You said `blah`, but I don't know anything about `blah`."

**Bonus chatter**: Other compilers produce slightly less confusing error messages by complaining about the incompleteness first.

```
// gcc
oops.cpp:7:15: error: invalid use of incomplete type 'struct blah'
oops.cpp:1:8: note: forward declaration of 'struct blah'

// clang
oops.cpp:11:15: error: argument type 'blah' is incomplete
oops.cpp:1:8: note: forward declaration of 'blah'

// icc
oops.cpp(11): error: cannot convert to incomplete class "blah"
```