The Code Using References

```
// a synchronization primitive...
bool compareXchgRef (ALPHA& alpha,
     int32_t& compare, int32_t& newVal)
{
    if (alpha.getNum () == compare) {
        alpha.setNumFromPtr (&newVal);
        return true;
    }
    return false;
}
```

Acts like an **ALPHA**.

Acts like an **int32_t**.

Acts like an **int32_t**.

References Allow a Choice of Argument Semantics

Finally, to call the functions...

```
compareXchgPtr (&a, &one, &two);
```

versus

```
compareXchgRef (a, one, two);
```

Notice that references **allow** a **programmer**
- to **redefine argument semantics**
- **on a per-argument basis**:
- some may be **call-by-value**, while others are **call-by-reference** (pointer).

Another Round of "Help Prof. Lumetta!"

I have a bug.

This loop seems to hang.

```
while (42 != i) {
    foo (i);
    x = bar (i);
    zap (i, x);
}
```

Can you help?

References are Easily Abused

As Stroustrup says, **any language can be misused**.
The **worst part** in my view:
- one can **change argument style**
- **without** generating **warnings at call sites** (code that uses the function)!

In other words,
- some code may assume that an argument doesn't change,
- but someone "redefines" the function
- so that it can change!

