Single-Argument Constructors Create Implicit Casts

```
class complex {
    // ...
public:
    complex (int32_t real_part);
    complex (double real_part);
    friend complex operator*
        (const complex& a,
         const complex& b);
}
```

Creates implicit cast from **`int32_t`** to **`complex`**.

Creates implicit cast from **`double`** to **`complex`**.

Use `explicit` to Avoid Creating Implicit Casts

The constructors enable the following:

```
complex c = 5; // cast from int32_t
complex d = 42.9; // from double
```

To **avoid creating implicit casts, add `explicit`** before the constructor.

Be aware that **casts can be chained**.

- Eliminating the **`double`** cast alone affects … almost … nothing:
- the compiler casts from **`double`** to **`int32_t`** to **`complex`**!

Operators Often Return Whole Instances

```
class complex
    // ...
public:
    complex (i
    complex (double real_part);
    friend complex operator*
        (const complex& a,
         const complex& b);
}
```

`friend` functions are in the global namespace, not in the class, so access control has no relevance.

Return type is the whole instance!

Instance Returned on Stack, Becomes Temporary in Caller

Why does the operator return an instance?

Other options?

- Pointer to automatic variable doesn't work.
- Dynamic allocation is more expensive.

So **an instance is returned on the stack**.

Back **in the caller**,

- the result is **a temporary**.
- Eventually, a destructor must be called.*
- Usually at the end of the statement.

*Except if a new variable is constructed from the temporary.

