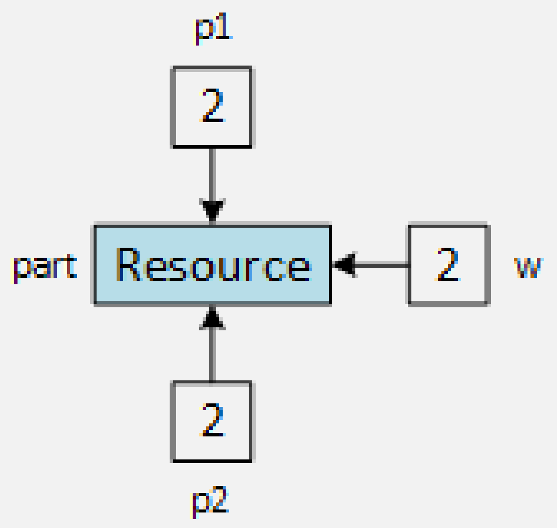
AN INTRODUCTION TO SMART POINTERS

Objects managing pointers

Delroy A. Brinkerhoff

C++ SMART POINTERS

- Classes wrapping raw pointers
  - The raw pointer is `private`
  - Access through `public` functions
  - Maintain a reference count
- Three kinds of smart pointers
  - `unique_ptr`
  - `shared_ptr`
  - `weak_ptr`



SMART POINTERS ARE TEMPLATE CLASSES

```
class part
{
    private:
        string name;
        int    id;
    public:
        part(string n) : name(n) {}
        ~part() { cout << "dtor\n"; }
        string get_name()
            { return name; }
};
```

```
int main()
{
    shared_ptr<part> p1 =
        make_shared<part>("Widget“, 1);
    shared_ptr<part> p2 = p1;
    weak_ptr<part> w = p1;
    cout << p1.use_count() << endl;
    return 0;
}
```

unique_ptr

```
unique_ptr<part> unique
    = make_unique<part>("Widget", 10);

cout << unique->get_name() << endl;

part* p = unique.release();

cout << p->get_name() << endl;
```

```
if (unique)
    cout << unique->get_name() << endl;
else
    cout << "unique is empty\n";

unique.reset(new part("Screw", 30));

if (unique)
    cout << unique->get_name() << endl;
else
    cout << "unique is empty\n";
```

shared_ptr

```
shared_ptr<part> shared =
    make_shared<part>("Bolt", 20);
shared_ptr<part> shared2 = shared;
shared_ptr<part> shared3 =
    make_shared<part>("Bolt", 20);

cout << "(1) " << shared->get_name() <<
    "  " << shared.use_count() << endl;
cout << "(2) " << shared2->get_name() <<
    "  " << shared2.use_count() << endl;
cout << "(3) " << shared3->get_name() <<
    "  " << shared3.use_count() << endl;
```

```
if (shared.unique())
    cout << "Unique\n";
else
    cout << "Shared\n";

shared.reset(new part("Screw", 30));

if (shared)
    cout << shared->get_name() << endl;
else
    cout << "shared is empty\n";
```

weak_ptr (1)

```
shared_ptr<part> shared = make_shared<part>("Gadget", 40);
weak_ptr<part> weak = shared;

cout << "(1) " << shared->get_name() << " " <<
    shared.use_count() << endl;
cout << "(2) " << weak.use_count() << endl;

shared_ptr<part> locked = weak.lock();
cout << "(4) " << shared.use_count() << "  " <<
    locked.use_count() << "  " << weak.use_count() << endl;
```

weak_ptr (2)

```
weak.reset();

if (weak.expired())
    cout << "weak unavailable" << endl;
else
    cout << weak.use_count() << endl;
cout << "(5) " << shared.use_count() << "  " <<
locked.use_count() << endl;
```