Inteligentny wskaźnik `std::weak_ptr`

dr inż. Ireneusz Szcześniak

jesień 2017

1 Wprowadzenie

Inteligentny wskaźnik `std::weak_ptr` pozwala na śledzenie obiektu, który jest zarządzany przez `std::shared_ptr`. Śledzenie nie daje nam gwarancji, że obiekt istnieje, ale daje nam możliwość ewentualnego dostępu, jeżeli zarządzany obiekt jeszcze istnieje.

```
#include <cassert>
#include <iostream>
#include <memory>

using namespace std;

struct A
{
  A()
  {
    cout << "ctor\n";
  }

  void
  saysomething()
  {
    cout << "Hi!\n";
  }

  ~A()
  {
    cout << "dtor\n";
  }
};

int
main()
{
  auto sp = make_shared<A>();
  weak_ptr<A> wp(sp);

  // Here the managed object exists.
  {
    shared_ptr<A> sp2(wp);
    sp2->saysomething();
    assert(sp2 != nullptr);
  }

  // Make the managed object go away.
  {
```

```
    auto sp2 = move(sp);
  }

  // Here the managed object is already gone.
  {
    shared_ptr<A> sp2 = wp.lock();
    assert(sp2 == nullptr);
  }
}
```

2 Zadanie

Napisać funkcję factory, która będzie fabryką obiektów klasy A. Fabryka powinna śledzić już stworzone obiekty i nie tworzyć ich ponownie, jeżeli ciągle istnieją.

3 Rozwiązanie zadania

```
#include <cassert>
#include <iostream>
#include <map>
#include <memory>

using namespace std;

struct A
{
  static int counter;

  int m_id;
  int m_unique;

  A(int id): m_id(id), m_unique(counter++)
  {
    cout << "ctor:␣" << m_id << ",␣" << m_unique << '\n';
  }

  ~A()
  {
    cout << "dtor:␣" << m_id << ",␣" << m_unique << '\n';
  }
};

int A::counter = 0;

auto
factory(int id)
{
  shared_ptr<A> sp;

  static map<int, weak_ptr<A>> cache;
  auto i = cache.find(id);

  if (i != cache.end())
    sp = i->second.lock();
```

```
  if (sp == nullptr)
    {
      sp = make_shared<A>(id);
      cache.insert(i, make_pair(id, sp));
    }

  return sp;
}

int
main()
{
  int unique;

  {
    auto sp1 = factory(1);
    auto sp2 = factory(1);
    unique = sp1->m_unique;
    assert(sp1->m_unique == sp2->m_unique);
  }

  auto sp1 = factory(1);
  assert(unique != sp1->m_unique);
}
```