C++11 中的智能指针

2016-10-15 c++ 991 words 2 mins read 12888 times read

C++11 中正式引入了智能指针 `unique_ptr` 、 `shared_ptr` 和 `weak_ptr` ，并推荐使用 `unique_ptr` 代替 C++03 中过时的 `auto_ptr` 。本文重点介绍前两种智能指针的用法。

什么是智能指针

智能指针是一类对象，虽然它们的行为表现像传统指针，却可以**管理**用 new 创建的对象。因此你不必担心什么时候用 delete 销毁对象，因为这些工作都将由智能指针帮你自动完成。智能指针中包含一个传统指针，指向模板类对象。因此智能指针可以指向任何类型的对象，只需在创建智能指针的时候将该类型作为模板参数传递给智能指针。

对于动态分配的对象，存在所有权问题。所谓所有权，是指该对象由哪些代码调用或销毁。当对象的创建和销毁处于不同代码段，并且所有权没有得到正确处理的时候，那么将导致出现内存泄漏或未定义行为（undefined behavior）。智能指针正是为了解决这类问题而引入的。

unique_ptr

`unique_ptr` 管理的对象只能被单个 `unique_ptr` 所独有，即两个 `unique_ptr` 不能指向同一个对象。它可以像传统指针一样使用：

```
unique_ptr<double> dp{new double};
*dp = 7;
```

它不需要使用 `delete` 销毁对象，当指针超出作用域时将自动销毁。因此不能使用非动态分配的内存来定义 `unique_ptr` ：

```
double d;
unique_ptr<double> dd{&d}; // 错误：dd的析构函数试图销毁d
```

可以用 `get` 方法获得裸指针（raw pointer）：

```
double* raw_dp= dp.get();
```

一个 `unique_ptr` 不能被赋值给另一个 `unique_ptr` ：

```
unique_ptr<double> dp2{dp};  // 错误：禁止拷贝构造
dp2 = dp;                    // 错误：禁止赋值
```

只能使用 `move` 方法，同时将所有权转移给新的 `unique_ptr` ：

```
unique_ptr<double> dp2{move(dp)}, dp3;
dp3= move(dp2);
```

上述代码中，被管理对象的所有权由 `dp` 转移给 `dp2` 再转移给 `dp3` ，最后对象由 `dp3` 管理， `dp` 和 `dp2` 都变成 `nullptr` 。

当函数返回对象为 `unique_ptr` 时，不需要显式调用 `move`：

```
unique_ptr<double> f()
{
    return unique_ptr<double>{new double};
}

int main(int argc, char *argv[])
{
    unique_ptr<double> dp3;
    dp3 = f();
    return 0;
}
```

shared_ptr

`shared_ptr` 是引用计数（reference counting）智能指针。其内部有一个引用计数器，用来记录有多少个 `shared_ptr` 指向被管理对象。所有指向同一被管理对象的 `shared_ptr` 共享该对象的所有权。当引用计数器为0时，说明没有任何智能指针指向被管理对象，此时对象将自动销毁。

和 `unique_ptr` 相反， `shared_ptr` 可以随意拷贝构造和赋值：

```
shared_ptr<double> f()
{
    shared_ptr<double> p1{new double};
    shared_ptr<double> p2{new double}, p3 = p2;
    cout << "p3.use_count() = " << p3.use_count() << endl;
    return p3;
}

int main(int argc, char *argv[])
{
    shared_ptr<double> p = f();
    cout << "p.use_count() = " << p.use_count() << endl;
    return 0;
}
```

上述代码中，当函数 `f()` 返回时， `p1` 指向对象的内存被回收， `p3` 指向对象的内存不回收，因为 `main` 函数中的 `p` 还在引用该对象。

参考资料

Gottschling, P. (2015). *Discovering Modern C++: An Intensive Course for Scientists, Engineers, and Programmers*. Addison-Wesley Professional.

Author wwzhao
LastMod 2016-10-15