OpenFOAM 中的 autoPtr

2016-10-22 OpenFOAM 511 words 2 mins read 12889 times read

autoPtr 类和 tmp 类一样，也是智能指针。头文件里面对 autoPtr 类的描述为：

> An auto-pointer similar to the STL auto_ptr but with automatic casting to a reference to the type and with pointer allocation checking on access.

CONTENTS

可见 autoPtr 和 std::auto_ptr 功能类似，是不支持引用计数的智能指针。

下面以 OpenFOAM-3.0.0 为例分析其源码实现，相关代码：

```
src/OpenFOAM/memory/autoPtr/autoPtr.H
src/OpenFOAM/memory/autoPtr/autoPtrI.H
```

成员变量

```
template<class T>
class autoPtr
{
    // Public data

        //- Pointer to object
        mutable T* ptr_;
    ...
```

autoPtr 类只有一个成员函数，即指向被管理对象的指针 `ptr_` 。

拷贝构造函数

```
template<class T>
inline Foam::autoPtr<T>::autoPtr(const autoPtr<T>& ap)
:
    ptr_(ap.ptr_)
{
    ap.ptr_ = 0;
}
```

拷贝构造函数将 `ap.ptr_` 赋值给 `ptr_` ，并清空 `ap.ptr_` 。拷贝构造函数的作用是：将被管理对象的所有权从 `ap` 转移到 `*this` 。因此执行完拷贝构造函数之后，无法再从 `ap` 访问被管理对象。

成员函数

autoPtr::ptr()

```
template<class T>
inline T* Foam::autoPtr<T>::ptr()
{
    T* ptr = ptr_;
    ptr_ = 0;
    return ptr;
}

```

`ptr()` 将清空 `ptr_` ，返回被管理对象的指针。因此执行完 `ptr()` 后， `*this` 不再拥有被管理对象的所有权。

autoPtr::reset(T* p)

```
template<class T>
inline void Foam::autoPtr<T>::reset(T* p)
{
    if (ptr_)
    {
```

```
        delete ptr_;
    }

    ptr_ = p;
}
```

`reset(T* p)` 将销毁原被管理对象，并使 `*this` 拥有对 `p` 的所有权。

操作符重载

赋值操作符

```
template<class T>
inline void Foam::autoPtr<T>::operator=(const autoPtr<T>& ap)
{
    if (this != &ap)
    {
        reset(const_cast<autoPtr<T>&>(ap).ptr());
    }
}
```

赋值操作符调用了 `ap.ptr()`，即收回 `ap` 对被管理对象的所有权，同时使用 `reset(T *p)` 获得 `*p` 的所有权。赋值操作符的作用类似拷贝构造函数，将被 `ap` 管理的对象的所有权转移给 `*this`。