参考： <http://blog.csdn.net/u012332679/article/details/60772201>

之前了解C++函数返回值类型时，首先是创建一个临时对象并将返回值拷贝构造给该临时对象，然后将临时对象返回并销毁临时对象，后参考上面的网址，发现临时对象并非一定会销毁。

例：程序C++11\_test10

class Test

{

public:

Test() : ptr\_(new int(3))

{

*cout* << "Construct address: "<< *hex* << this << *endl*;

}

Test(Test& t) : ptr\_(new int(\*t.ptr\_))

{

*cout* << "Copy construct from address: " << *hex* << &t << " to address: " << this << *endl*;

}

Test& operator=(Test& t)

{

*cout* << "Assign: " << *endl*;

if (ptr\_ != nullptr)

delete ptr\_;

ptr\_ = new int(\*t.ptr\_);

return \*this;

}

~Test()

{

delete ptr\_;

ptr\_ = nullptr;

*cout* << "Destruct address: " << this << *endl*;

}

public:

int\* ptr\_;

};

Test GetTest() {

Test t;

return t;

}

第一种情况：

int main(int argc, char\* argv[])

{

*cout* << "call function GetTest() start." << *endl*;

GetTest();

*cout* << "call function GetTest() end." << *endl*;

return 0;

}

输出为：

call function GetTest() start.

Construct address: 008FFC00

Copy construct from address: 008FFC00 to address: 008FFC30

Destruct address: 008FFC00

Destruct address: 008FFC30

call function GetTest() end.

从输出可以看出：

调用GetTest()时，首先创建对象t，地址为008FFC00，在返回时利用t拷贝构造临时对象，地址为008FFC30，然后析构t，函数调用完成返回后析构掉临时对象008FFC30，可见函数返回后，临时对象被销毁了。

第二种情况：

*cout* << "call function GetTest() start 2." << *endl*;

Test t1 = GetTest();

*cout* << "call function GetTest() end 2." << *endl*;

*cout* << "t1 address: " << &t1 << *endl*;

输出为：

call function GetTest() start 2.

Construct address: 004FF814

Copy construct from address: 004FF814 to address: 004FF920

Destruct address: 004FF814

call function GetTest() end 2.

t address: 004FF920

Destruct address: 004FF920

第二种情况，有对象接收GetTest()的返回值，可以看出临时对象地址004FF920在函数返回时并未销毁，接收对象t1的地址也是004FF920，可见临时对象在函数返回时并未销毁，而是在main结束时才销毁的。

上述两种情况下，都函数调用时都至少执行了一次构造函数，一次拷贝构造函数，一次析构函数，原因就是由于临时对象的存在。如果希望减少这些调用，可如下：

Test GetTest1()

{

return Test();

}

*cout* << "call function GetTest1() start." << *endl*;

GetTest1();

*cout* << "call function GetTest1() end." << *endl*;

*cout* << *endl*;

*cout* << "call function GetTest1() start 2." << *endl*;

Test t2 = GetTest1();

*cout* << "t2 address: " << *hex* << &t2 << *endl*;

*cout* << "call function GetTest1() end 2." << *endl*;

输出为：

call function GetTest1() start.

Construct address: 00D5FA38

Destruct address: 00D5FA38

call function GetTest1() end.

call function GetTest1() start 2.

Construct address: 010FF704

t2 address: 010FF704

call function GetTest1() end 2.

Destruct address: 010FF704

从输出可以看出，GetTest1()返回时，并未拷贝构造临时对象，原因是语句：

return Test();本身就是返回一个临时对象。